Electronic Thesis or Dissertation Chapter Summary Application

ChapterClassSumm Group:
  Jackson, Miles
  Zhao, Yingjie

Course: CS 4624 Multimedia, Hypertext and Information Access
  Instructor: Edward A. Fox
  Client: Bipasha Banerjee

Virginia Tech,
Blacksburg, VA 24061
5/7/2024
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table of Tables</td>
<td>3</td>
</tr>
<tr>
<td>Table of Figures</td>
<td>4</td>
</tr>
<tr>
<td>1. Abstract</td>
<td>5</td>
</tr>
<tr>
<td>2. Introduction</td>
<td>7</td>
</tr>
<tr>
<td>2.1 Problem</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Motivation</td>
<td>7</td>
</tr>
<tr>
<td>2.3 General Approach</td>
<td>8</td>
</tr>
<tr>
<td>3. Requirements</td>
<td>9</td>
</tr>
<tr>
<td>4. Design</td>
<td>11</td>
</tr>
<tr>
<td>4.1 Login System</td>
<td>11</td>
</tr>
<tr>
<td>4.2 Upload a PDF</td>
<td>12</td>
</tr>
<tr>
<td>4.3 Results Screen</td>
<td>13</td>
</tr>
<tr>
<td>5. Implementation</td>
<td>16</td>
</tr>
<tr>
<td>5.1 Segmentation</td>
<td>16</td>
</tr>
<tr>
<td>5.1 Application</td>
<td>16</td>
</tr>
<tr>
<td>Built Upon Architecture</td>
<td>17</td>
</tr>
<tr>
<td>Front-End</td>
<td>17</td>
</tr>
<tr>
<td>Back-End</td>
<td>20</td>
</tr>
<tr>
<td>5.3 What was delivered</td>
<td>22</td>
</tr>
<tr>
<td>6. Testing/Evaluation/Assessment</td>
<td>23</td>
</tr>
<tr>
<td>6.1 Testing</td>
<td>23</td>
</tr>
<tr>
<td>6.2 Evaluation</td>
<td>23</td>
</tr>
<tr>
<td>6.3 Assessment</td>
<td>26</td>
</tr>
<tr>
<td>7. Users' Manual</td>
<td>28</td>
</tr>
<tr>
<td>7.1 Use Cases</td>
<td>28</td>
</tr>
<tr>
<td>7.2 Tutorial</td>
<td>28</td>
</tr>
<tr>
<td>8. Developer's Manual</td>
<td>31</td>
</tr>
<tr>
<td>8.1 File Inventory</td>
<td>31</td>
</tr>
<tr>
<td>8.2 Setup</td>
<td>32</td>
</tr>
<tr>
<td>8.3 Up and Running</td>
<td>33</td>
</tr>
<tr>
<td>9. Lessons Learned</td>
<td>34</td>
</tr>
<tr>
<td>9.1 Timeline/schedule</td>
<td>34</td>
</tr>
<tr>
<td>9.2 Problems and Solutions</td>
<td>35</td>
</tr>
<tr>
<td>Problem 1: File Sharing</td>
<td>35</td>
</tr>
<tr>
<td>Problem 2: Text Display</td>
<td>35</td>
</tr>
<tr>
<td>Problem 3: Storing Large Texts</td>
<td>36</td>
</tr>
<tr>
<td>Problem 4: CSV to Database</td>
<td>36</td>
</tr>
<tr>
<td>Problem 5: Display Layout</td>
<td>36</td>
</tr>
<tr>
<td>Problem 6: Session State</td>
<td>37</td>
</tr>
</tbody>
</table>
9.3 Future Work....................................................................................................................... 37

10. Acknowledgements............................................................................................................ 39
    10.1 Bipasha Banerjee......................................................................................................... 39
    10.2 Edward A. Fox........................................................................................................... 39

11. References........................................................................................................................ 40
Table of Tables

Table 1: Segmentation Results Verification Table.................................................. 16
Table 2: Etds Table Schema.................................................................................. 20
Table 3: Metadata Table Schema.......................................................................... 21
Table 4: Timeline and Schedule............................................................................. 35
Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Designed Login Screen Wireframe</td>
<td>11</td>
</tr>
<tr>
<td>1.2</td>
<td>User Account Menu Wireframe</td>
<td>12</td>
</tr>
<tr>
<td>1.3</td>
<td>PDF Upload Screen Wireframe</td>
<td>13</td>
</tr>
<tr>
<td>1.4</td>
<td>Summarization Results Wireframe</td>
<td>14</td>
</tr>
<tr>
<td>1.5</td>
<td>ETD Chapter Summarization Screen Wireframe</td>
<td>15</td>
</tr>
<tr>
<td>2.1</td>
<td>Web Interface Homepage</td>
<td>18</td>
</tr>
<tr>
<td>2.2</td>
<td>Homepage Error Message Example</td>
<td>18</td>
</tr>
<tr>
<td>2.3</td>
<td>Summarization Results Singular Summarization Model</td>
<td>19</td>
</tr>
<tr>
<td>2.4</td>
<td>Summarization Results from Multiple Models</td>
<td>19</td>
</tr>
<tr>
<td>2.5</td>
<td>Metadata Excel File</td>
<td>21</td>
</tr>
<tr>
<td>2.6</td>
<td>Snapshot of the MySQL Metadata Table</td>
<td>22</td>
</tr>
<tr>
<td>3.1</td>
<td>Summarization Without Upload Error</td>
<td>24</td>
</tr>
<tr>
<td>3.2</td>
<td>Incorrect Naming Format Error</td>
<td>24</td>
</tr>
<tr>
<td>3.3</td>
<td>Metadata Not Found Error</td>
<td>25</td>
</tr>
<tr>
<td>3.4</td>
<td>MySQL Connection Error (Front-End)</td>
<td>25</td>
</tr>
<tr>
<td>3.5</td>
<td>MySQL Connection Error (Back-End)</td>
<td>26</td>
</tr>
<tr>
<td>3.6</td>
<td>Non-PDF Uploaded Error</td>
<td>26</td>
</tr>
<tr>
<td>4.1</td>
<td>Application UML Diagram</td>
<td>28</td>
</tr>
<tr>
<td>2.1</td>
<td>Web Interface Homepage</td>
<td>29</td>
</tr>
<tr>
<td>2.3</td>
<td>Summarization Results Singular Summarization Model</td>
<td>29</td>
</tr>
<tr>
<td>2.4</td>
<td>Summarization Results from Multiple Models</td>
<td>30</td>
</tr>
</tbody>
</table>
1. Abstract

The US corpus of Electronic Theses and Dissertations (ETDs), partly captured in our research collection numbering over 500,000, is a valuable resource for education and research. Unfortunately, as the average length of these documents is around 100 pages, finding specific research information is not a simple task. Our project aims to tackle this issue by segmenting our sample of 500,000 ETDs, and providing a web interface that provides users with an application that summarizes individual chapters from the previously segmented sample.

The first step of the project was to verify that the automatic segmentation process, performed in advance by our client, could be relied upon. This required each team member to analyze 50 segmented documents and verify their integrity by confirming that each chapter was correctly identified and separated into a PDF. During this process, we noted any peculiarities, to identify recurring issues and improve the segmentation process. The rest of our time and effort went into creating an efficient web interface that would allow users to upload ETD chapters and display said chapter’s summary and classification results.

We were able to complete a web interface that allows a user to upload an ETD chapter PDF from the sampled ETD database and view the summary of the PDF along with all of the metadata (author, title, publication date, etc.) of the associated ETD. Additionally, the group verified approximately 60 of the automatically segmented documents and detailed any errors or peculiarities thoroughly. Our group delivered both the web interface as a GitHub repository and an Excel spreadsheet detailing the complete results of our segmentation verification process.

The interface was designed to be used in aiding research on ETDs. Although this application won’t be available publicly, researchers may use it privately to assist with any ETD research projects they participate in.

The web interface uses Streamlit, which is a Python framework for web development. This was the first time anyone in the group had used Streamlit, so we had to learn each feature that we used, which caused quite a few issues. However, quickly searching and accessing the metadata database, which was originally an Excel sheet with 500,000 entries, posed the biggest threat to the usability of our interface. Luckily, we were able to solve all issues through the use of API documentation, our client, Bipasha Banerjee, and our extremely helpful instructor, Professor Edward A. Fox.

In terms of technical skills, we have learned how to operate a Streamlit web interface as well as how to use MySQL. However, we also learned a few life lessons. Firstly, do not use the first tool available when attempting to solve a solution. It is wise to take extra time to search for the best
tool for a given situation instead of wasting time compensating for using the wrong tool. Secondly, life happens without regard and without warning, but the best move is to reanalyze the situation and push forward to complete the work that must be done.
2. Introduction

2.1 Problem
Electronic theses and dissertations (ETDs) are scholarly reports that document the research of graduate students worldwide. They are formatted in such a way that makes them ideal for machine archives and worldwide retrieval. For this reason, their popularity has skyrocketed and ETDs have been solidified as a commonplace medium to document graduate research today.

By design, each ETD is a comprehensive collection of all information surrounding the research topic. Additionally, all methods used in the project, any complications that were encountered during the process, and the solutions to those complications, must be included in the report as well. This abundance of information can be extremely useful to those who wish to verify the integrity of the research, redo the study, or perform research on an adjacent topic. However, to those who are interested in using a very specific part of the work to reference for their publications, the amount of information can be overwhelming and difficult to sift through.

That being said, students and researchers often are required to support their works with other scholarly reports and, therefore, many refer to ETDs. Combined with the difficulty of navigating these ETDs, we have an issue where a large number of students and researchers struggle to complete the same task of analyzing ETDs for specific pieces of data, again and again.

2.2 Motivation
As students ourselves, we know the struggle of parsing through a 100-page document to find a single sentence and/or paragraph. Furthermore, sometimes that sentence/paragraph is to be used as evidence for a single point of an argument in which you are required to have at least three points in total. Doing so can result in an extremely exhausting experience performing what should have been a relatively easy task. For that reason, being able to give back to the community that has guided us for the last 4 years is a great honor. Through this project, not only will undergraduate students be able to manage school more easily, but graduate students will have the satisfaction of their work being used and cited more. However, the application would only be of use if the chapter segmentation process worked correctly.

Motivation for segmentation:
- This is needed to get clear chapter boundaries from the entire ETD. Useful for creating chapter summaries and classification labels.

Motivation for the application:
- The application helps users visualize the chapter summary and labels. This helps to mitigate having to read long documents without any indication of what is the desired portion of interest. We show the chapter-related content to help users with this problem
2.3 General Approach
The main goal was to couple the already developed deep learning models with an easy-to-use web interface. The Streamlit Python framework was created with the idea of supporting machine learning models, so the task of plugging deep learning models into the back-end is a simple one. For the front-end UI, we kept things clear and concise. We weren’t concerned with making the web interface beautiful (although that is a bonus), but we wanted the interface to be effective and not confusing.
3. Requirements

During the Initial Inception of the project contract, our client put most of the focus on manually segmenting the ETDs and using that data as a training set for a deep learning model to further improve its performance. The subsequent phase of developing the UI was mostly included as a ‘stretch’ goal or a goal that could be completed if the other phases of the project were completed quickly enough. This was because Yingjie was the only member of the group at the time.

Once Miles was added to the project group, the project was split into two parts: Summarization and Classification. Unfortunately, Yingjie, the member assigned to the classification portion of this project, faced several hardships in his personal life and was unable to complete his project work during the semester. Therefore, only the summarization function of the project was required for submission. That being said, Yingjie still contributed heavily to the design of the project and still deserves recognition for his efforts.

The requirements were later changed due to the advancement of the chapter segmentation deep learning models. This allowed for segmentation to be done automatically and our new task was to verify the integrity of the segmentation job. This granted us more time to work on the front-end and half of our stretch goals were moved into the main scope of the project. In the end, our required deliverables were a completed chapter segmentation results validation report in the form of an Excel spreadsheet, and a functional user interface that allowed users to summarize ETD chapters.

Therefore, the finalized requirements and official deliverables for this project were:
- Verification of segmentation results from 100 ETDs
- Create a user interface that displays chapter summary and classification results.

Chapter segmentation results verification: Our group was assigned 100 ETDs (50 per person) and their chapter segmentation results. Additionally, we were also tasked with verifying the integrity of these chapter segmentation results. More specifically, we verified that each ETD file contains the correct number of segmented PDFs that match the number of chapters in the ETD. The next step was to correctly identify the chapter boundaries of each segmented chapter and note down any errors encountered in that process. Lastly, we noted down anything special that we noticed while verifying the integrity (e.g., if a chapter contains a summary/discussion segment that provides great data for the summarization learning model, if the references are incorrectly attached to the last chapter which would adversely affect the summarization model, etc.). All of these observations were recorded in an Excel spreadsheet, and each entry is uniquely identified by the corresponding ETD’s ID number which represents its position in the file system managed by our client.
Functional User Interface: For this deliverable, the logistics were not rigidly defined. We were to create a **functional** UI and that was the only requirement. We could use any software (that is reasonable for this application) and any system as long as the interface works. However, the specific ‘function’ that qualified our interface as functional was the ability of our system to allow the uploading of a PDF and display the results of the summarization of the PDF to the user accurately. A previous group [2] started work on this interface using the Streamlit Python framework, so we decided to build upon their work to accomplish the task.
4. Design

Our interface design was separated into 3 main parts.

4.1 Login System

The user login system was designed to increase the security of our application as well as provide the ability to save uploaded PDFs even after the user closes the website. It would be undesirable for the user to upload the same PDF each time they close the application/tab (intentionally or accidentally). If we can keep track of the user’s account, we can save recently uploaded PDFs so that the user can easily revisit relevant data with less hassle. As shown in Figure 1.1, the application will offer a very basic sign-in function using usernames and passwords. Additionally, a “Forgot Password” function will be available for account recovery in case of user forgotten credentials.

![Designed Login Screen Wireframe](https://www.ClassificationAndSummarization.org)

**Figure 1.1: Designed Login Screen Wireframe**

Figure 1.2 shows that the account and sign-in/sign-out options will always be available in the top right corner of the header of every page.
4.2 Upload a PDF
As outlined in Figure 1.3, once the user logs in, they can either access their most recently uploaded PDF or upload a new one. The most recent PDFs will be displayed by their title and date of upload in a list fashion on the right side of the screen (encapsulated in a sidebar). The “Upload a PDF” button will be centered on the remaining portion of the screen which is otherwise blank.
4.3 Results Screen
Once a PDF is uploaded, it is then pushed through to the deep learning models to be classified and summarized. As shown in Figure 1.4, right above the summaries is a section displaying the metadata of the file (author, date of publication, organization, etc.). The middle portion of the screen will have collapsible text boxes that each correspond to a chapter. When enlarged, the text boxes show the chapter summaries. At the bottom of the page, a home button will take you back to the “Upload a PDF” page where you can upload another PDF file or access a previously uploaded document.
The summarization model will work for individual ETD chapter uploads. As shown in Figure 1.5, the resulting summarization screen will only display the chapter’s summary without any collapsible text boxes.
Figure 1.5: ETD Chapter Summarization Screen Wireframe
5. Implementation

5.1 Segmentation

Our group was assigned chapter segmentation results for 100 ETDs (divided into 50 per member). We were instructed to fill out an Excel file with the same schema as detailed in Table 1.

<table>
<thead>
<tr>
<th>ETDID</th>
<th>University</th>
<th>Department</th>
<th>Correct</th>
<th>Comments</th>
<th># of Correct Chapters</th>
<th>Total # of Chapters</th>
<th>To be Done By</th>
<th>Time taken</th>
</tr>
</thead>
</table>

Table 1: Segmentation Results Verification Table

The fields were used as follows:

- **ETDID** - The ID of the ETD the results corresponded to.
- **University** - The university the ETD was published from.
- **Department** - The department of the University the ETD was published under.
- **Correct** - Yes or No field. Overall determination if the segmentation was correct or not.
- **Comments** - Where we were instructed to point out any particular aspects of the segmentation that stood out (e.g. “References are attached to concluding chapter”, “Chapter 1 incorrectly identified as 2nd page of table of contents”, “Chapter 1,2,3 include discussion sections”, etc.). Chapters with their own discussion sections were particularly important to note as those discussion sections could serve as a valuable reference to check our own chapter summaries against.
- **# of Correct Chapters** - The number of chapters within the ETD that were segmented correctly
- **Total # of Chapters** - Total # of Chapters within the ETD. Used in comparison with # of correct chapters
- **To be Done By** - Explained which group member was to be responsible for this ETD segmentation
- **Time taken** - Time taken to perform the verification. Measured in minutes

Although Yingjie had to cease working part way during the project, he was able to complete 6 of the ETD segmentation verifications. All other members of the group completed all 50 of their assigned verification tasks. This resulted in a total of 56 completed ETD segmentation verifications being delivered.

5.1 Application
Unfortunately, we were unable to implement everything that was designed for the application. Here are the features that weren’t transferred from the design to the final product:

- User Log-in function
- Uploaded ETD chapter PDF tracking (Recently uploaded PDFs section)
- The ability to upload a complete ETD and receive the results pertaining to each chapter in collapsible text boxes
- The classification of ETD chapter PDFs

However, the rest of the features were implemented successfully.

Built Upon Architecture

When our client, Bipasha Banerjee, briefed us on the requirements and expectations for the web interface, she mentioned that a previous team had already developed a skeleton for the interface [2] using Streamlit. She then mentioned that we were able to continue developing the previous team’s code or start anew. After some research, our group decided to continue using Streamlit, as the platform was designed to accommodate the machine learning integration, which was necessary for our client. However, upon further inspection of the code, we realized that we would have to redesign the entire interface and only keep the underlying code that processed uploaded PDFs.

Front-End

When starting the interface, you are greeted with the homepage as shown in Figure 2.1. The homepage includes a brief description of what the website’s function is and how to begin using the application. Below the explanation is an area to upload a PDF, and buttons for summarization and classification. For summarization, you can choose which machine learning model you want to use through a dropdown menu.
Users can upload ETD chapter PDFs via the upload file box, but only one file is allowed to be uploaded at any given time. Furthermore, classification has not been implemented, so clicking the classification button at any point or clicking the summarization button without uploading a file results in an error message which is shown in Figure 2.2.

Once a user has uploaded an ETD chapter file, they may click the summarize button which will summarize the chapter PDF with the currently selected summarization model. As shown in
Figure 2.3, the summarization results are then displayed along with all of the relevant metadata associated with the submitted ETD chapter.

![Figure 2.3: Summarization Results Singular Summarization Model](image)

For just one summary for a piece of text, the entire summary is automatically displayed as displayed in Figure 2.3, however, if multiple summaries exist, then each summary will be contained within collapsible text boxes that state the summarizer model that created the summary, as shown in Figure 2.4.

![Figure 2.4: Summarization Results from Multiple Models](image)
Back-End

The project uses a single MySQL database that contains two tables: the etds table and the metadata table. The schema for the etds and metadata tables are shown in Table 1 and Table 2, respectively.

<table>
<thead>
<tr>
<th>file_name</th>
<th>etd_id</th>
<th>chapter_id</th>
<th>text_path</th>
<th>summ_path</th>
<th>summ_model</th>
</tr>
</thead>
<tbody>
<tr>
<td>[String - 256 characters max]</td>
<td>[Integer - Id number of ETD associated with chapter]</td>
<td>[String - Chapter name]</td>
<td>[String - path of text file]</td>
<td>[String - path of summarization file. Primary key for this table]</td>
<td>[String - machine learning model used for summarization]</td>
</tr>
</tbody>
</table>

Table 2: Etds Table Schema

<table>
<thead>
<tr>
<th>table_id</th>
<th>etd-id</th>
<th>title</th>
<th>author</th>
<th>advisor</th>
<th>year</th>
<th>abstract</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Integer - Entry # in Excel file]</td>
<td>[Integer - Id number of ETD. Primary key for this table]</td>
<td>[String - title of ETD]</td>
<td>[String - author of ETD]</td>
<td>[String - Advisor assigned to ETD]</td>
<td>[Integer - Year published]</td>
<td>[String - Abstract for the ETD]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>university</th>
<th>degree</th>
<th>uri</th>
<th>department</th>
<th>discipline</th>
<th>language</th>
<th>schooltype</th>
</tr>
</thead>
<tbody>
<tr>
<td>[String - University the ETD was published from]</td>
<td>[String - the degree under which the author published the ETD]</td>
<td>[link to the ETD]</td>
<td>[String - the department in which the author published the ETD]</td>
<td>[String - the discipline under which the author published the ETD]</td>
<td>[String - the language of the ETD]</td>
<td>[String - Type of school from which the ETD was published]</td>
</tr>
</tbody>
</table>
Each ETD chapter that is uploaded has its text and summaries stored in text files in the project directory while the *ets* table stores the path to these files. This implementation serves to reduce stress on the database, by preventing the storage of huge amounts of text. Additionally, since we store the summary path as our primary table key instead of the summary itself, we can check if a summarization exists by checking its existence via the file path. If we stored the summary itself in the database, we would have to create the summary first and then check if the generated summary matched the content of a summary already in the database, which would decrease the runtime speed.

The biggest detriment to runtime speed, however, came from accessing data from the metadata Excel file. Originally, the metadata was left in the Excel file and accessed via the native Python CSV library, Lib/csv.py [7]. However, due to the size of the metadata file, all queries through this method timed out. We solved this issue by ingesting the metadata into the *metadata* table we mentioned earlier. Figure 2.5 shows the original Excel file that we needed to access for the metadata and Figure 2.6 shows an example entry from the *metadata* table we transitioned to.

---

Table 3: Metadata Table Schema

<table>
<thead>
<tr>
<th>Name</th>
<th>University</th>
<th>Degree</th>
<th>City</th>
<th>State</th>
<th>Country</th>
<th>ETD Number</th>
<th>Title</th>
<th>URL</th>
<th>ETD Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Doe</td>
<td>University of Illinois</td>
<td>PhD</td>
<td>Urbana</td>
<td>IL</td>
<td>USA</td>
<td>123456789</td>
<td>The Impact of Climate Change on Agriculture</td>
<td><a href="https://example.com/">https://example.com/</a></td>
<td>REGULAR</td>
</tr>
<tr>
<td>Jane Smith</td>
<td>Stanford University</td>
<td>Masters</td>
<td>Palo Alto</td>
<td>CA</td>
<td>USA</td>
<td>987654321</td>
<td>Development of Sustainable Energy Systems</td>
<td><a href="https://example.com/">https://example.com/</a></td>
<td>REGULAR</td>
</tr>
</tbody>
</table>

---

Figure 2.5: Metadata Excel File

---

Figure 2.6: Example Metadata Entry from *metadata* Table
Figure 2.6: Snapshot of the MySQL Metadata Table

5.3 What was delivered

As a group, we were initially expected to submit:

- Verification of segmentation results from 100 ETDs
- A user interface that displays chapter summary and classification results.

However, we submitted:

- Verification of segmentation results from 56 ETDs
- A user interface that displays chapter summary results.

Although we did not deliver everything that was asked of us, I am proud that we were able to deliver what we did after facing plenty setbacks during the project runtime.
6. Testing/Evaluation/Assessment

6.1 Testing

Due to the interface not being available publicly, all testing was performed internally by our group and our client, Bipasha Banerjee. Using the specifications and requirements she initially crafted for the interface, Bipasha attempted to use the website as she initially envisioned and reported any issues.

In regards to finding bugs and/or trying to break our code, the team performed every action possible at each stage in the program and worked to prevent any bugs and glitches that were found from this practice.

6.2 Evaluation

Bugs/Errors Found:

- If a user accesses the summarization results page before uploading any ETDs (via the sidebar) they will be met with a non-fatal ‘Variable not initialized’ error due to multiple summarization variables not having values. This error is shown in Figure 3.1.
- When uploading a PDF file that doesn’t begin with “[Integer].pdf”, such as “451654.pdf_chapter_1.pdf”, there is a type error when entering the assumed ‘ETD’ into the etds table. The code automatically assumes the naming format of all submissions will follow the example above. This error is shown in Figure 3.2.
- If the ETD ID is not in the metadata file, a type error is thrown when trying to access the data from the metadata table. To display the metadata information, the code indexes through the queried metadata, but Python does not allow for the indexing of NoneType variables. This error is shown in Figure 3.3.
- If the connection to the Streamlit server is interrupted, the connection to the MySQL database fails after the connection to Streamlit is re-established without redeploying the entire application. As shown in Figure 3.4, the error causes the interface to load infinitely and doesn’t indicate what issue caused the interface to crash. However, by looking at the console snapshot in Figure 3.5, you can see that the MySQL connection failed and caused a fatal error within the application.
- If a non-PDF file is uploaded, a “PDFSyntax Error” is thrown when attempting to read the text content from the file. This error is shown in Figure 3.6.
Figure 3.1: Summarization Without Upload Error

Figure 3.2: Incorrect Naming Format Error
Figure 3.3: Metadata Not Found Error

Figure 3.4: MySQL Connection Error (Front-End)
6.3 Assessment

The data from the *metadata* table is ingested beforehand and has already been thoroughly checked by the client and her project team, so no errors arise from interacting with this table. However, when it comes to inserting user-uploaded data into the *etds* table, which is created and maintained by the code during operation, there are no safeguards against errors induced by user actions. If the user doesn’t upload a specific file type, the file’s name doesn’t follow the naming format expected, or the file corresponds to an ETD that is supported by the *metadata* table, each one of these scenarios will result in an error. These errors can be addressed within the code, but it is important to document that these errors still exist at the time of writing this report.

Beyond user actions, the possibility of losing connection to the MySQL database that contains the tables essential to the application, the *etds* table and *metadata* table, represents a massive
weak point for the application. I am unsure if the behavior will persist when running the application on a server or if any solution can be implemented within the code itself, but, if this problem persists unsolved, it will be the most limiting factor of use.
7. Users' Manual

7.1 Use Cases

The application user flow is very simple for this project because the application is only used to summarize ETD chapters. Users will either be on the homepage to upload an ETD chapter PDF or on the summarization results page to view the summaries of the uploaded PDF chapter.

![Application UML Diagram](image)

Figure 4.1: Application UML Diagram

7.2 Tutorial

Once the user enters the application, they must upload an ETD chapter to review the summarization results of said chapter. As shown in Figure 2.1, you can drag and drop or browse your computer files to upload the desired ETD chapter PDF. The user must also select their desired summarization model from the dropdown menu next to the “Summarize” button.
On the summarization results page shown in Figure 2.3, the application displays the metadata associated with the ETD chapter PDF they uploaded on the homepage. For this particular chapter, there is only a summary for a singular summarization model, Model-1, so the summary is displayed automatically without any further prompts. Once the user is done viewing the summarization results, the user may exit the application or head back to the homepage to upload another ETD chapter via the “Back” button.
If there are multiple summaries from multiple different summarization models available for a single chapter, each different summary will be contained in a collapsible text box that’ll be titled the same as the summarization model associated with the summary contained in the text box. Of course, the “Back” button is still available under this circumstance. This functionality is shown in Figure 2.4.

Figure 2.4: Summarization Results from Multiple Models

8.1 File Inventory

- `__pycache__` directory - All files in this directory are auto-generated and maintained from running the application. **Do not** alter these files.
  - `Experimenter_Home.cpython-311.pyc` - Cache file associated with Experimenter_Home python file
  - `mysql_connection.cpython-311.pyc` - Cache file associated with mysql_connection python file

- `.streamlit/config.toml` file
  - Streamlit page config file. Currently used to set the site theme to “light”.

- `etd_text_files` directory
  - This directory is used to store all text files from uploaded ETD chapters and summaries generated from said ETD chapters. Chapter texts and summaries are stored to lower application costs by preventing using the summarizer models to create summaries that already exist.

- `helpers` directory
  - `__pycache__` directory - All files in this directory are auto-generated and maintained from running the application. **Do not** alter these files.
    - `pdf_reader.cpython-311.pyc` - Cache file associated with pdf_reader python file
    - `mysql_connection.cpython-311.pyc` - Cache file associated with mysql_connection python file
    - `sidebar.cpython-311.pyc` - Cache file associated with sidebar python file
  - `mysql_connection.py` file
    - This Python file handles the connection to the MySQL database used for the project. I am currently using my personal MySQL account for the project. However, if a future team or the client wishes to change which account is used, the username, password, and host location fields in this file will need to be changed to connect to the new MySQL account. You can make a new account and download MySQL if needed here: [https://dev.mysql.com/downloads/mysql/](https://dev.mysql.com/downloads/mysql/). This file also configures the indices of relevant data in each MySQL table for easier database access in the rest of the project.
  - `pdf_reader.py` file
    - Inherited from the previous team [2]. Used to read text from PDF files.
  - `README.md` file
- Markdown readme that explains how to import the helpers into any project file. Helpers are all of the Python files in the `helpers` directory (the directory this README.md file is contained in).
  - `sidebar.py` file
    - Python configuration file for the Streamlit sidebar.
- `pages` directory - Contains all of the non-homepage website pages.
  - `summarization.py` file
    - Python file responsible for the entirety of the summarization results page. More specifically, this page uses Streamlit to create, organize, and display the key elements of the summarization results page: metadata and summaries.
- `.gitignore` file
  - Details of what files are ignored when pushing the project to Git.
- `environment.yml` file
  - YML file that can be used to automatically create a virtual Python environment that can run this project.
- `etds_500k.csv` file
  - Excel file which holds metadata for all 500k of the currently supported ETDs.
- `etdsDB.sql` file
  - MySQL file was used to create the underlying databases used in the project.
- `Experimenter_Home.py` file
  - Python file responsible for the entirety of the homepage. More specifically, this page uses Streamlit to create, organize, and display the key elements of the homepage: file upload, summarizer model selection, and summarization button.
- `README.md` file
  - Readme file that explains how to run the application on a local machine.
- `requirements.txt` file
  - Details all of the required software for this project to operate. Can use the `pip install -r requirements.txt` command to automatically install dependencies.

### 8.2 Setup

**OS Recommendation:** Linux preferred (Windows/MacOS accepted)

To run the application, these Python libraries must be installed in the Python environment running the project:

- `streamlit` - version 1.25.0
- `Pandas` - version 2.0.3
- `tabulate` - version 0.9.0
- `streamlit_ace` - version 0.1.1
- `PyPDF2`
- `transformers`
- `scikit-learn`
• python-dotenv
• pdfplumber
• streamlit_extras
• mysql-connector-python

Once you have set up your Python environment, you can use the requirements.txt file to automatically install all dependencies. Simply run the command:

```
pip install -r requirements.txt
```

8.3 Up and Running

Once you’ve installed the software packages listed in Section 8.1, follow these instructions to access the application:


2. Download MySQL (https://dev.mysql.com/downloads/mysql/) and configure Root user access for your computer. Please change passwords from default to ensure secure access.

3. Create a database and tables as specified in etdsDB.sql.
   a. Open a terminal on your computer and navigate to the project folder.
   b. Open MySQL with this command: "mysql --local-infile -uRootUsername -pRootPassword". (local-infile tag allows you to populate a database with local data)
   c. If you still get an error that prevents you from ingesting data locally, you will have to run this command while inside MySQL: SET GLOBAL local_infile=1;
   d. In MySQL, type "create database etdsDB;" then "use etdsDB;" and finally "source ./etdsDB.sql;"
   e. Change 'user', 'password', and/or 'database' fields in helpers.mysql_connection.py as necessary

4. Create a virtual environment and install all the packages mentioned in requirements.txt.

5. Run the code using the “streamlit run Experimenter_Home.py” command. This will start the streamlit server on the default 8501 port. To change the port, add server.PORT <port_number> to the run command.

6. Provide an ETD chapter in the file upload area and enjoy!

For any issues, reach out to Miles Jackson (mjaxon@vt.edu).
9. Lessons Learned

9.1 Timeline/schedule

<table>
<thead>
<tr>
<th>Date(s)</th>
<th>Milestone</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feb 1, 2024 - Feb 15, 2024</td>
<td>Presentation 1</td>
<td>Dataset Collection and Annotation Setup. Select 100 ETDs for the segmentation dataset. Validate segmentation results. Begin the annotation process. Review the existing codebase and select the technology stack.</td>
</tr>
<tr>
<td>Feb 16, 2024 - Feb 29, 2024</td>
<td>Validation Completion</td>
<td>Continue annotating chapter boundaries for the selected ETDs. Aim to complete the validation of the segmentation process. Deliver the first version of the application. Implement the functionality to accept chapter PDFs as input and display classification and summarization results.</td>
</tr>
<tr>
<td>Mar 1, 2024 - Mar 15, 2024</td>
<td>Prototype testing, Presentation 2, Interim Report</td>
<td>Continue the development of the UI and testing.</td>
</tr>
<tr>
<td>Mar 16, 2024 - Mar 31, 2024</td>
<td>Integration and Testing</td>
<td>Integrate the trained models with the UI. Ensure the system can process input PDFs and display the results accurately. Conduct thorough testing of the system. Add test cases. This includes testing for bugs,</td>
</tr>
</tbody>
</table>
Table 4: Timeline and Schedule

9.2 Problems and Solutions

Problem 1: File Sharing

**Issue:** Initially, we had an issue with file sharing. This was not particularly with the client but within the group. Only one member had access to the implementation and work completed so far, but that same member was unreachable.

**Solution:** Thereafter, all work was uploaded to a git repository so that all members may work on their own time and upload any results remotely.
Problem 2: Text Display

**Issue:** Originally, all of the text on the interface was displayed using a method called streamlit.text. This method did not work well with long bodies of text and did not allow for any content customization (bolding, italics, etc.).

**Solution:** After doing some research, I found another text-displaying method called streamlit.textArea. This method displayed large amounts of text correctly but was meant for textual input instead of output. Disabling the input functionality caused the text to appear slightly translucent.

The next method used was called streamlit.code. This method behaved very similarly to streamlit.textArea, but displayed text with 100% opacity. The only issues that remained were the lack of text customization and the size of the display box that accompanied the text.

Finally, we found streamlit.write. This method allowed for textual display with text customization, 100% opacity, and no surrounding display boxes. This allowed for a more intuitive and appealing interface appearance which pleased both our group and the client.

Problem 3: Storing Large Texts

**Issue:** Storing large amounts of text (entire ETD chapters ) in a MySQL database resulted in some of the data being lost due to storage constraints.

**Solution:** Instead of storing the text itself in the MySQL database, we opted to store the text in a separate text file and store the paths to those files in the database. This increased performance as well as prevented data from being lost.

Problem 4: CSV to Database

**Issue:** Accessing the metadata CSV file that contained 500,000 entries was extremely slow and costly.

**Solution:** Instead of accessing the CSV file directly from the code, we entered all of the data into a MySQL database which allows for more efficient accessing and searching of data.
Problem 5: Display Layout

**Issue:** Elements in Streamlit are arranged in rows from top to bottom, meaning that each subsequent element is shown under the previous element. This leaves a lot of empty space, which negatively impacts the site's appearance.

**Solution:** We discovered a method called streamlit.columns that divides the next row(s) into multiple columns to allow for elements to be placed next to each other.

Once the elements were placed horizontally, we noticed that the elements are top aligned, meaning that the top of each element starts in the same place. Initially, this doesn’t sound like much of an issue, but this resulted in our buttons being misaligned based on how long the label above them was. This fact resulted in the website looking a bit messy.

To remedy this issue, we had to learn and use a bit of HTML to override Streamlit’s native configuration and align the elements how we wanted.

Problem 6: Session State

**Issue:** Streamlit doesn’t allow for the intuitive passing of variables between pages (Python files).

**Solution:** The first attempt at passing variables resulted in importing the pages wherever their variables would be necessary, but that approach failed. When importing other pages, Streamlit took that as an instruction to recreate the imported page on top of the current page. Unfortunately, the variables did not carry over anyway. The old variables were reset as soon as the page was navigated away from and the new variables were instantiated to ‘None’.

Luckily, we found the streamlit.sessionState dictionary, which allows for data to be saved for an entire session. This one feature completely delivered what we were looking for.

9.3 Future Work

Much work remains before the project leads to a finished product. Firstly, there needs to be more safeguards in place to account for erroneous user actions. Furthermore, it would be beneficial for all future users and project teams to move the application from local machines to a server. This would centralize any work efforts, keeping errors and performance consistent for all users/developers moving forward.
Additionally, the amount of supported ETDs can be increased. Out of the entire set of ETDs available in the USA, only 500,000 are supported within this application. Although 500,000 is not a small number by any metric, if we’re able to extend the supported ETDs to much more, the scope of our application would be increased significantly.

Lastly, one of the biggest issues is the lack of summarization integration into the project. The summarization models have not been integrated, so all of the summaries are filled with dummy text. This means that, at the time of this report, there is no use of any machine learning models in the project. The current placeholder method should allow for the summarization models to be integrated seamlessly, but there is no way to know for sure as is.
10. Acknowledgements

10.1 Bipasha Banerjee
Ph.D. student and GRA, bipashabanerjee@vt.edu

Our deepest gratitude towards our client. She has helped us at every step of the way and shows us how much she cares about the project and our well-being too.

10.2 Edward A. Fox
CS4624 Professor, fox@vt.edu

Much appreciation for our professor. Professor Fox has supported us with any help or wisdom he could provide. We would not have this opportunity without him.
11. References


