VIRGINIA TECH

Team 4: Final Presentation

Under the guidance of Dr. Fox
SMEs: Bipasha Banerjee, Sareh Ahmadi

Team: Abhilash Neog, Aditya Shah, Deepak Nanjundan, Deval Srivastava, Dharneeshkar Jayaprakash, Kaushik Ganesan

Blacksburg, VA 24061
Dec. 19, 2022

AGENDA

- Milestones
- Overall Flow
- Segmentation
- Parse and Clean
- Summarization
- Classification
- Pipeline Integration
- User Interface
- Deliverables

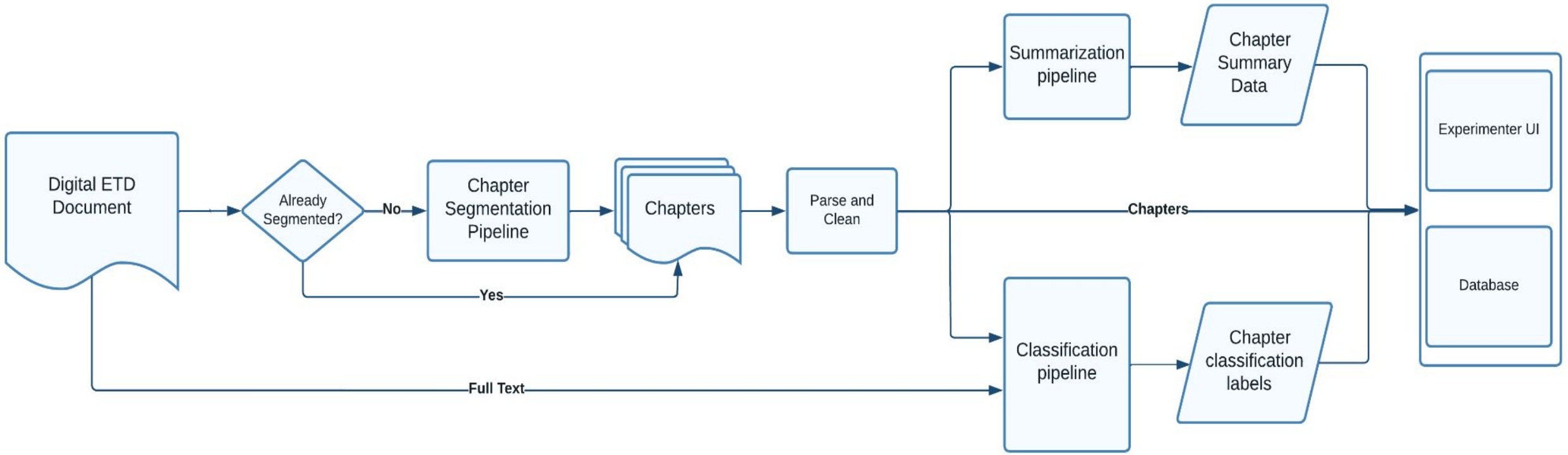
MILESTONES

USER STORIES

01	USER STORY 1	• As an experimenter, I want to segment the ETD into chapters. The entire ETD PDF is the input and the generated chapter PDFs are the output.
02	USER STORY 2	• As an experimenter, I want to provide a chapter text and obtain summarized version of the chapter.
03	USER STORY 3	• As an experimenter, I want to provide a chapter text and obtain the discipline labels of the chapter.
04	USER STORY 4	• As an experimenter, I want to access all the above mentioned services with a User Interface.

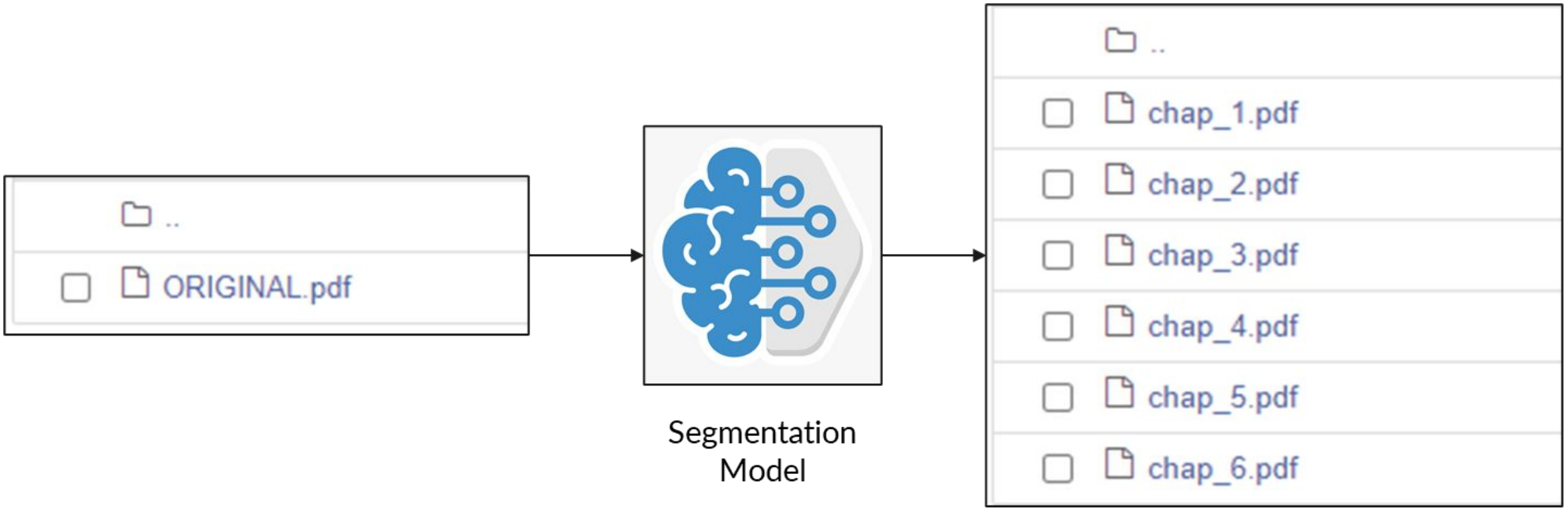
OVERALL FLOW

OVERALL FLOW DIAGRAM



SEGMENTATION

SEGMENTATION



SEGMENTATION-METHODOLOGY

VIRGINIA TECH

Input ETD	Page images & texts	Model Prediction	Chapter Boundary	Saving chapter PDFs
• The input to the segmentation pipeline is a single ETD PDF. • Currently the system supports only digital documents.	• The given ETD is split into page images and the text is extracted for each page. • Input to the model is page images and page texts.	For each page, the model predicts a classification label. • 0: “FM PAGE”, • 1: “FMSTART” • 2: “CH START” • 4: “CH PAGE” • 3: “EM START” • 5: “EM PAGE”	• **Approach 1:** Traverse the prediction array and find the page numbers of one “CH START” to the next one and save it as a chapter. • **Approach 2:** Traverse the prediction array and subarrays when more than 6 consecutive “CH PAGE” are saved as a chapter.	The generated chapter PDFs are saved as digital objects in Team 1’s database and file system.

SEGMENTATION-PROGRESS

- Packaged the code and set up in team 1’s container.
- Completed running segmentation on 5000 digital ETDs.
- Stored segmentation results in the file system and database.
- Containerized the service.
- Added segmentation service to team 5’s workflow automation (Experimenter page).

SEGMENTATION-FUTURE WORK

- Experiment Chapter Captor approach
- Increasing dataset size
- Hyper parameter tuning
- Unit testing

CLEAN AND PARSE

CLEAN AND PARSE

Methodology:

- Parses the chapter PDF.
- Removes figures and tables.
- Saves cleaned text into .txt file.
- **Library Used:** pdfplumber.

Clean and Parse as a service:

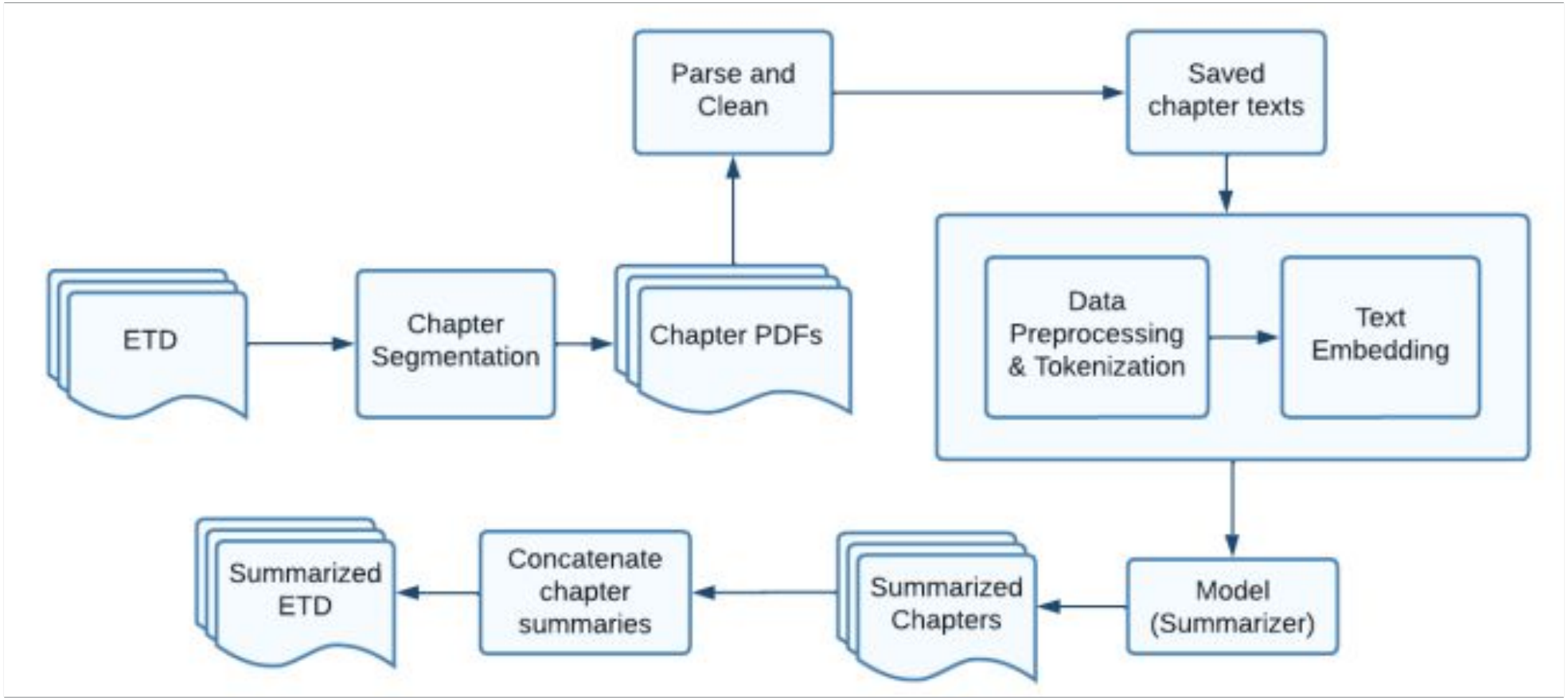
- Cleaned chapter text is an input for summarization and classification pipelines.
- We deployed a standalone service to parse and clean the script and store it in the database so that this process is not repeated.

CLEAN AND PARSE-FUTURE WORK

- Improve our clean and parse script so that it also removes equations.
- The output of pdfplumber is sometimes erratic, i.e., it doesn't convert its character dictionary back to ASCII.
- To avoid junk characters in the output of cleaned text, we can either manually convert the characters which weren't converted or else use some other library to parse the PDF.

SUMMARIZATION

SUMMARIZATION - FLOW



SUMMARIZATION-METHODOLOGY

SUMMARIZATION PIPELINE

- Sets the required parameters - model, summary length, etc.
- Passes a chapter text as input to the pipeline.
- Summarizes the chapter text using the model selected.
- Generates a summary JSON.

MODELS SUPPORTED

- Extractive models - TextRank, LexRank, LSA.
- Abstractive models - Hugging Face Transformer Models.
- Model used for generating summaries to be stored in DB - TextRank.

SUMMARIZATION-MODEL OUTPUTS


```
{
    "summarisation": "In this paper, Theobald discusses the importance of high-order arithmetic in solving large problems. He
argues that it is necessary to be able to solve larger problems when the problem involves multiple orders of magnitude and/or
multiple processes running simultaneously. For example, if a particle collides with an object at a fixed point in space , then the
resulting product must be divided into smaller portions for each order of magnitude. This results in very large sums being needed
to be solved. To illustrate how important these sums are, statistics are used to calculate the cross sections of products produced
by different types of reactions. One example is the production or absorption of hydrogen using proton beams. By measuring the
amount of energy released during the process, one can determine what part of the system needs to be shut down as soon as possible
so that there will not be too much of a build-up",

 "algorithm_used": "pszemraj/bigbird-pegasus-large-K-booksum"
}
```


Above: Transformer Summary; Below: TextRank Summary


```
{
    "summarisation": "Chapter 2 High order QCD and resummation 2.1 When is NLO not enough? In section 1.3, we have discussed
briefly the analytic NLO calculation of the full inclusive DIS and DY cross sections. However, in many cases, the NLO pQCD com-
putation turns out not to be enough. This is, for example, often the case at LHC where the Higgs boson production has to be
distinguished from the background. A computation beyond the NLO is needed also when the NLO corrections are large and higher-order
calculation permit us to test the convergence of the perturbative expan- sion. In figure 2.1 the total cross section of the
production of the Higgs boson at LHC [36] is plotted and we note convergence in going from LO to NLO and to NNLO. This is the case
for example for the rapidity DY distributions at Tevatron (shown in figure 5.4) and at the fixed-target experiment E866/NuSea
(shown in figure 2.3). The agreement Figure 2.1: Total cross section for the Higgs boson production at LHC at (from bottom to top)
at LO, NLO, NNLO in the gluon fusion channel [36]. 21 22 High order QCD and resummation Figure 2.2:
DYrapiditydistributionforprotonanti-protoncollisionsatTevatronat(frombottom to top) LO, NLO, NNLO, together with the CDF data
[37]. Figure 2.3: DY rapidity distribution for proton proton collisions at fixed-target experiment E866/NuSea at (from bottom to
top) LO, NLO, NNLO, together with the data [19, 38]. with the data of figure 5.4 has represented an important test of the NNLO
splitting functions [29, 30]. Calculations beyond the NLO can be important also in processes which involve large logaritms when
different significant scales appear. (57,77)of section 1.3, we see that there are contributions that become large when z \u2192 1
from the quark-antiquark channel in the DY case and from the quark channel for",

 "algorithm_used": "textrank"
}
```

SUMMARIZATION - CHALLENGES

- Model Evaluation - Given most of the ground-truth data consists of newly generated summaries, the evaluation results would be biased toward abstractive models.

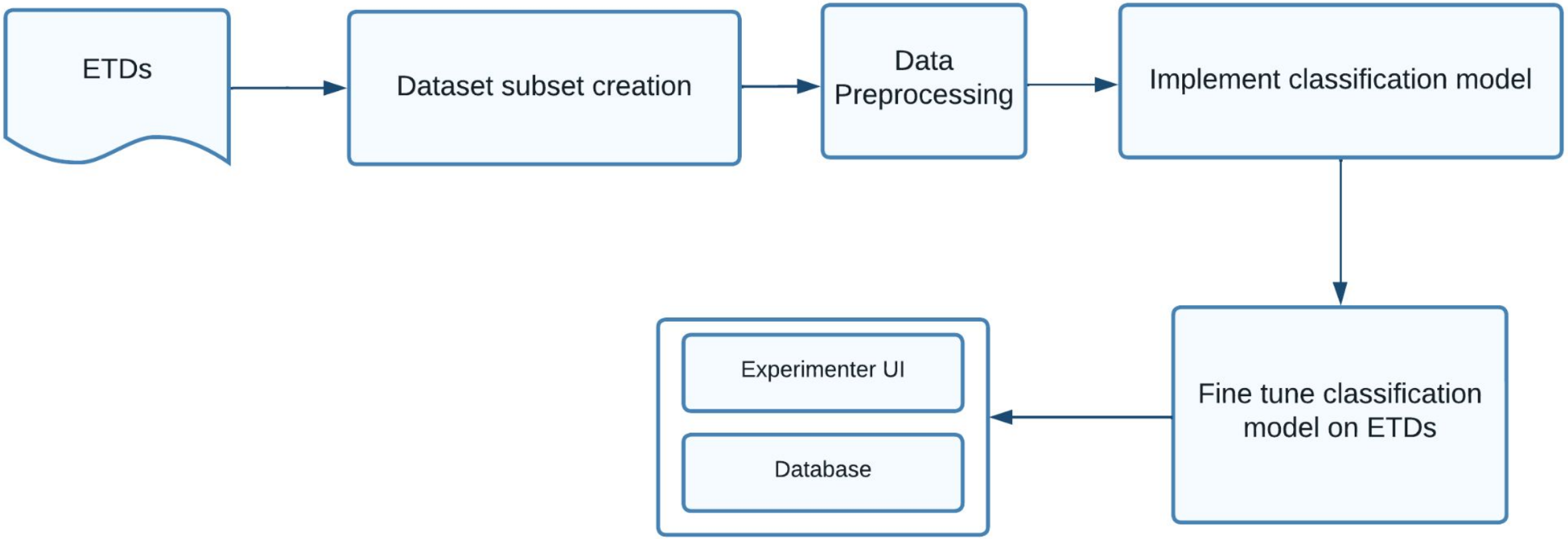
- Time - Transformer models (like Big Bird) need around seven to ten minutes to generate a single chapter summary compared to TextRank (that takes a few seconds). Hence, TextRank was chosen for summary generation for 5k ETDs.

SUMMARIZATION - FUTURE WORK

- Generating human-like summaries using abstractive models (transformer) - the current pipeline supports transformer models.
- Fine-tuning of pre-trained transformer models on ETD-relevant dataset.
- Exploring summary generation at multiple granularity level - section, paragraphs.

CLASSIFICATION

CLASSIFICATION FLOW



CLASSIFICATION-DATASET

- Ran the segmentation pipeline and collected chapters from around 1500 ETDs.
- The segmentation pipeline yielded 4000 chapters in total.
- Chapter-by-chapter PDF files were parsed and cleaned using the standalone Clean and Parse service.
- For a few chapters, the parsed output was not ideal because of flaws in the pdfplumber library.
- After eliminating the subpar chapter texts, we had 3742 chapters left with which we fine-tuned the models.

CLASSIFICATION-DATASET

- The “Discipline” feature in the metadata was used to generate the classification labels for the chapters in the ETDs.
- When the discipline labels were not available we used the department field as the label for the chapter.
- We utilized a training set to test set ratio of 85% to 15% to train and evaluate the models that were trained.
- Models were fine tuned on a training set and evaluated on a test set taken from a dataset of over 3500 chapters more than what was used for our previous evaluation.
- The chapter classification model trained can predict 27 different classes.

CLASSIFICATION- BERT and SciBERT

Implementation:

- Used BERT to generate representation and classify the chapter text.
- Also, trained SciBERT to classify the chapters.
- Selected top two predictions of the models trained as output.
- Since we hypothesized that chapter text would be longer than 512 tokens on average, we have also trained BERT and SciBERT on Text Rank summaries generated by our summarization team.
- However, average length of chapter text was less than 512 tokens.
- Since the probability of the chapter text being less than 512 tokens is remote, further assessments will be needed in the future.
- Models trained on summaries did not give us expected boost in performance.

Models Used: BERT-base and SciBERT

CLASSIFICATION- LongFormer

Implementation:

- Since we hypothesized that chapter text would be longer than 512 tokens on average, we have trained Longformer on chapter text as an alternative to BERT and SciBERT trained on chapter summaries.
- As average length of chapter text was less than 512 tokens, the performance of Longformer wasn’t great when compared to BERT and SciBERT.
- For smaller texts, Longformer was highly inefficient and took a longer time to train.

CLASSIFICATION- SVM AND RANDOM FOREST

Implementation:

- Experimented with traditional ML approaches to set a baseline.
- Replicated Palakh's experiments:
  - Apply lemmatization, stop word removal, etc.
  - Generate embeddings for the text using Doc2Vec/Fasttext.
  - Apply models on the embeddings to classify.
  - Experimented by running the code on PQDT dataset [1].

Models Used: SVM, Random Forest
Performance was not satisfactory on the PQDT dataset collected by Palakh [1].

[1] Palakh Mignonne Jude. "Increasing Accessibility of Electronic Theses and Dissertations (ETDs) Through Chapter-level Classification". June 2020, MS thesis, Computer Science, Virginia Tech, Blacksburg, VA 24061, http://hdl.handle.net/10919/99294.

CLASSIFICATION- TRAINING FLOW

- We have developed our model training and implementation code in PyTorch and Hugging Face.
- Overall the training pipeline follows the following steps:
  - Read the Chapter or summary text dataset with target labels.
  - Split the dataset into training and testing.
  - PyTorch DataLoader is set up which facilitates training.
  - Tokenizer is called within the DataLoader and it generates input text tokens.
  - The model is initialized and a pre-trained model is loaded in memory.
  - Optimizers and LR schedulers are initialized for training.
  - Model is trained and prints the losses periodically.
- Finally, once the model is trained, we save the final weights and perform evaluation.

CLASSIFICATION- EVALUATION

Model Name	F-1 Score (%)	Accuracy (%)	Trained on (Chapter/Summary)
BERT	56.28	64.75	Chapter
SciBERT	77.5	80.77	Chapter
SciBERT	67	70.89	Summary
BERT	54.41	63.01	Summary

Discussion of Results

- Results clearly show that, across the board, classification models trained on chapters are better than models trained on summaries.
- Since the average text in the chapter was less than 512 tokens in most cases, loss of information was not taking place.
- After applying summarization on chapter text we condensed the information even more leading to loss of some accuracy.
- It should also be noted that the summaries were generated via TextRank based method so were not of the highest quality.
- Keeping these points in mind we used SciBERT based on chapters for processing the 5k documents.
- At last we also tested the Longformer models but their performance was significantly worse than BERT based models.

CLASSIFICATION- PROGRESS

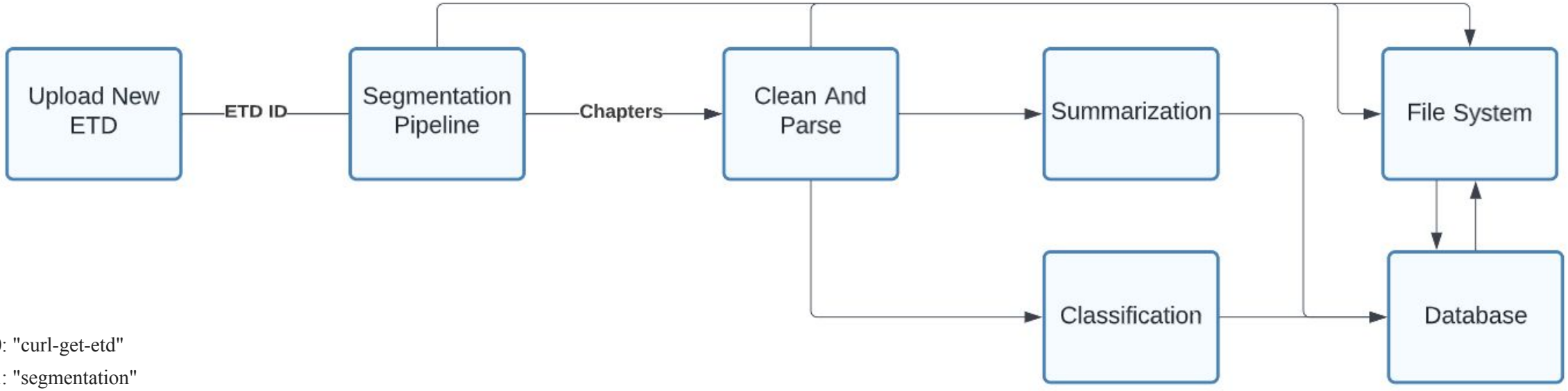
- We have successfully stored the results for 5000 ETDs subset in the database.
- We have used SciBERT model to store the results.
- Classification results were stored in the objects table under the type “cleaned_text”.
- Both the model used (name) and two labels predicted by the classification model were stored in the database.
- Classification service was also integrated with workflow automation and user interface.

CLASSIFICATION- FUTURE WORK

- Since the probability of the chapter text being less than 512 tokens is remote, further assessments will be needed in the future.
- We have successfully prepared a pipeline to train and evaluate models for metadata classification but did not execute it due to lack of time.
- We can also use chapter classification models to generate labels for metadata.
- Knowledge distillation of larger models trained (BERT, SciBERT) will greatly reduce the inference time and training time - if the models are retrained in the future.
- Classifying intent behind the citations can be very useful.
- Finding the purpose of a citation helps to determine the publication's influence and direct users towards additional readings.
- Citation Intent can be classified as Background, Methodology and Result Comparison similar to SciCite.
- Tuning methodologies such as Prompt Tuning can be explored for all models.

PIPELINE INTEGRATION

PIPELINE INTEGRATION- WORKFLOW AUTOMATION



0: "curl-get-etd"
1: "segmentation"
2: "curl-save-chapters"
3: "clean-chapters"
4: "summarization"
5: "classification"
6: "save-summarization"
7: "save-classification"
8: "save_cleaned_chapters"

ACCESSING TEAM 4 SERVICES

Segmentation:

- Pull the image: docker pull dharneeshkar/segmentation:0.4
- To run the segmentation pipeline use: python segmentation.py --input-path ./data/ --output-path ./out

Summarization:

- Pull the image: docker pull dharneeshkar/summarization:0.9
- To run the summarization pipeline use: python summarization.py --input-path ./data/ --output-path ./out

Classification:

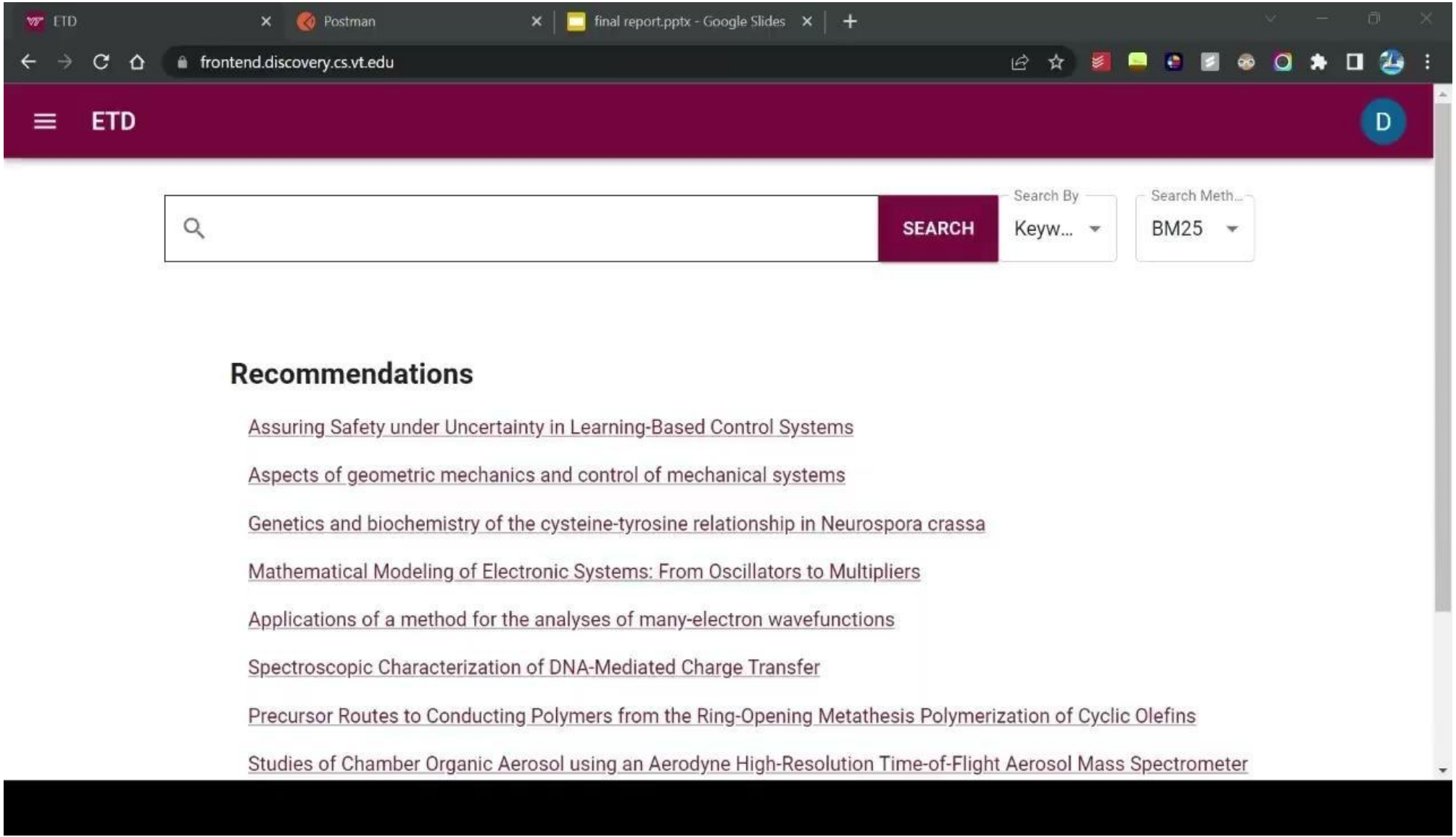
- Pull the image: docker pull dexuiz/classification:0.0.3
- To run the classification pipeline use: python3 classifier.py --input-path ./data/chap.txt --output-path ./out

Clean and Parse:

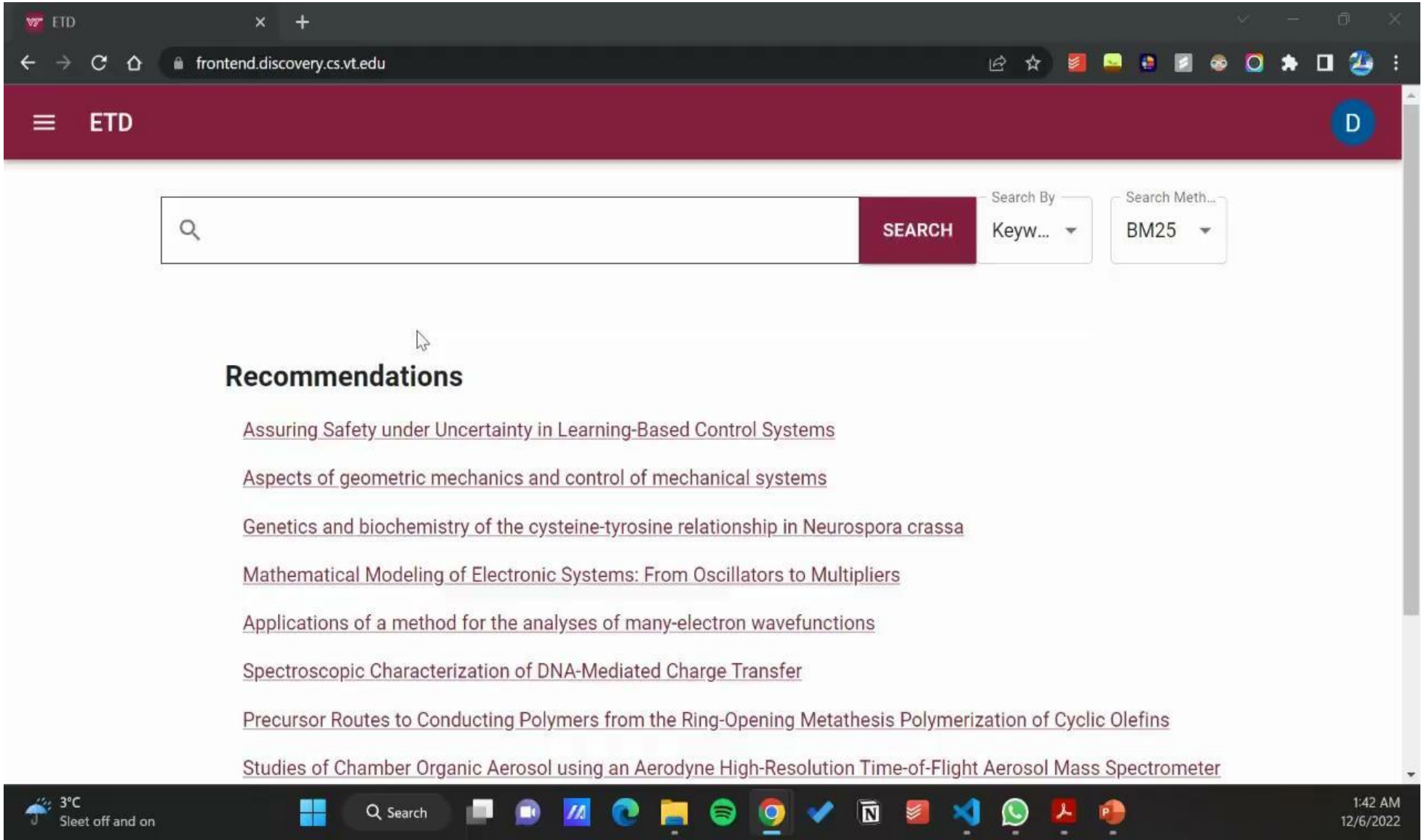
- Pull the image: docker pull dharneeshkar/parse_and_clean:0.1
- To run the clean and parse pipeline use: python parse_and_clean.py --input-path ./data/chapters --output-path ./out

USER INTERFACE

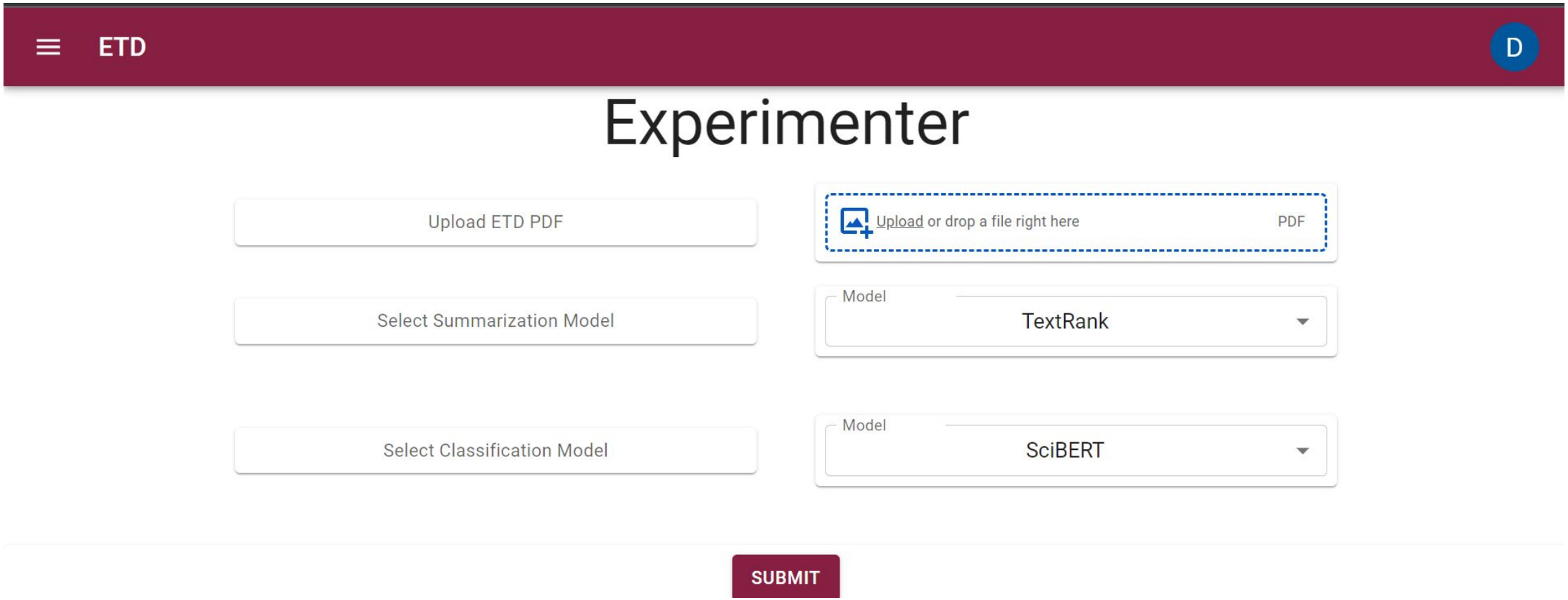
EXPERIMENTER PAGE



DOCUMENT VIEW PAGE



USER INTERFACE



USER INTERFACE

Title	Classification Label	Chapter Summary
Chapter 1	Electrical and Computer Engineering, Biomedical Engineering	1 Chapter 1 Introduction 1.1 Autonomous Driving System Autonomous driving (AD) is the task of driving a vehicle without any human intervention in urban and highway conditions. If we step back and stare at the existing autonomous driving system shown in Figure 1.1, there is a similar hand-designed aspect of the autonomous driving system that leads to suboptimal performance. 7 Chapter 2 End-To-End Driving Models Robust perception-action models should be learned from training data with diverse visual appearances and realistic behaviors, yet current approaches to deep visuomotor policy learning have been generally limited to in-situ models learned from a single vehicle or simulation environment. We advocate learning a generic vehicle motion model from large scale crowd- sourced video data, and develop an end-to-end trainable architecture for learning to predict a distribution over future vehicle egomotion from instantaneous monocular camera observations and previous vehicle state. Our model incorporates a novel FCN-LSTM architecture, which can be learned from large-scale crowd-sourced vehicle action data, and leverages available scene segmentation side tasks to improve performance under a privileged learning paradigm. Given a large-scale driving video dataset, an end-to-end FCN-LSTM network is trained to predict multi-modal discrete and continuous driving behaviors. We formulate the problem as learning a generic driving model/policy; our learned model is generic in that it learns a predictive future motion path given the present agent state. Our driving model is akin to a language model, which scores the likelihood of character or word sequences given certain corpora; our model similarly is trained and evaluated in terms of its ability to score as highly likely the observed behavior of the held out driving sequence. Finally, we report experimental results confirming that “privileged” training with side task (semantic segmentation) loss learns egomotion prediction tasks faster than from motion prediction task loss alone1. This
Chapter 2	Computer Science, Electrical and Computer Engineering	To solve this problem, we propose a Perception- Logic Network that unsupervisely learns the scene factors and combine them with a logic network. 4.2 Related Work Autonomous Driving The seminal work of Pomerleau [Pomerleau, 1989a] proposed to use learning method and neural networks to automatically learn driving policies from the data. 4.3 The Perception-Logic Network In this section, we will describe the scene conditioned driving problem formulation, and our proposed Perception-Logic Network to solve this problem. The unsupervised scene conditioned driving problem is defined as: given a set of driving demonstrations {(s ,a)}, and the corresponding set of scene conditioned symbolic rules t t f (x ,··· ,x),i = 1···r, to learn a driving agent a = π(s) that exhibits the scene conditioned i 1 k driving behavior. Note that the diversity loss is applied on every node of the logic network, including both the input unsupervised learned scene factors and the intermediate logic variable during the computation of the logic network. We name the diversity loss on the input logic factors as Gating Network Diversity Loss, and name the diversity loss on the intermediate computations as Switch Weight Diversity Loss. The row with Model G is the ground-truth label for weather Gating Network (WT), Temporal Gating Network (TP), Town Gating Network (TN) and output driving speed (SP) in unit km/h. For weather gating, 1 represents clear and 0 represents rainy; for temporal gating, 1 represents noon and 0 represents sunset; for town gating, 1 represents rural area and 0 represents urban area and for speed, 10 represents slow driving at 10km/h and 20 represents fast driving at 20km/h, which is the speed of experts’ driving behaviour in our training data. The row with Model C is the output of our double branch network without any diversity loss

Rows per page: 100 1–2 of 2

Workflow Status: Completed

DEMONSTRATION VIDEOS

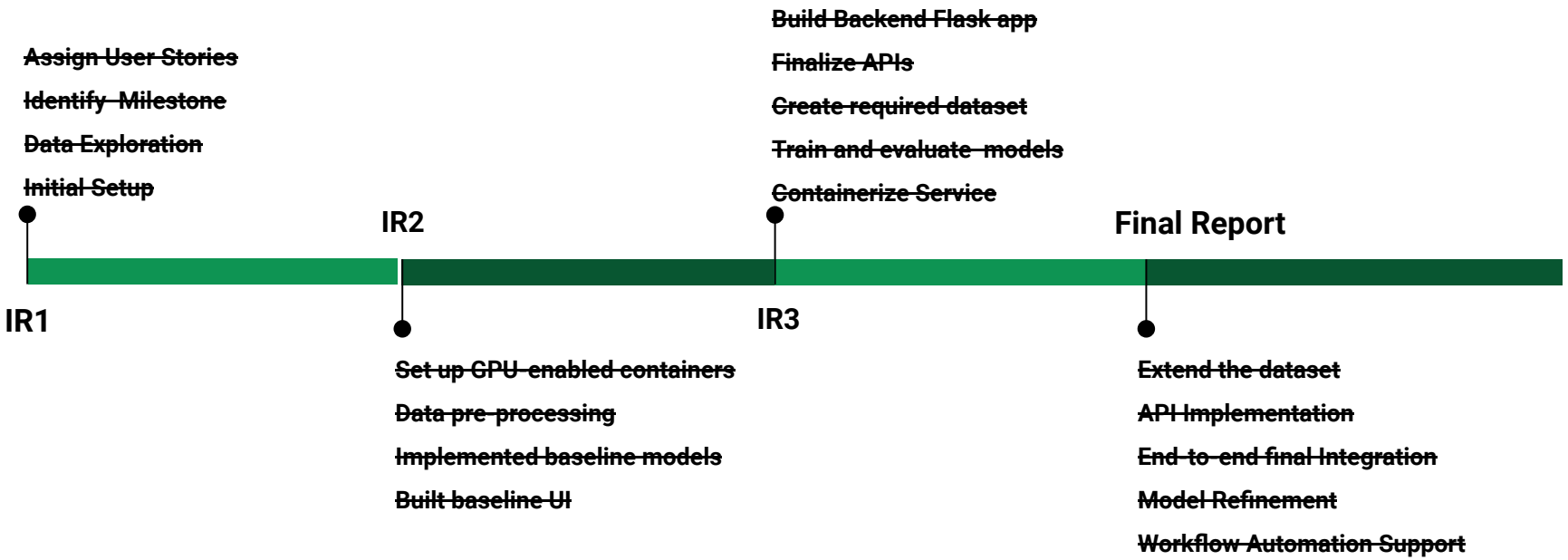
- File team4_documentView_demo.mp4 - Document View Page of User Interface Demo Video
- File team4_experimenter_demo.mp4 - Experimenter Page User Interface Demo Video

FINAL DELIVERABLES

FINAL DELIVERABLES

Deliverable	File System/ DB	Can be Retrieved by
Segmentation	File System	https://team-1-flask.discovery.cs.vt.edu/v1/etds/<etd-id>/objects?type=chapter
Cleaned Chapter Text	File System	https://team-1-flask.discovery.cs.vt.edu/v1/etds/<etd-id>/objects?type=cleaned_text
Summarization	DB	https://team-1-flask.discovery.cs.vt.edu/v1/etds/<etd-id>/objects?type=cleaned_text
Classification	DB	https://team-1-flask.discovery.cs.vt.edu/v1/etds/<etd-id>/objects?type=cleaned_text
UI Integrating all the subsystems mentioned	-	-

FINAL DELIVERABLES



THANK YOU

VIRGINIA TECH