# Final Report
# CS 5604: Information Storage and Retrieval

Team 3: Objects & Topics
Nila Masrourisaadat, Raj Sahu, Alan Devera,
Nirmal Amirthalingam, Chenyu Mao
Subject Matter Expert: Aman Ahuja

January 17, 2023
Instructed by Professor Edward A. Fox

Virginia Polytechnic Institute and State University
Blacksburg, VA 24061

## Abstract

The CS 5604: Information Storage and Retrieval class (Fall 2022), led by Dr. Edward Fox, has been assigned the task of designing and implementing a state-of-the-art information retrieval and analysis system that will support Electronic Theses & Dissertations (ETDs). Given a large collection of ETDs, we want to run different kinds of learning algorithms to categorize them into logical groups, and by the end, be able to suggest to an end-user the documents which are strongly related to the one they are looking for. The overall goal for the project is to have a service that can upload, search, and retrieve ETDs with their derived digital objects, in a human-readable format. Specifically, our team is tasked with analyzing documents using object detection and topic models, with the final deliverable being the Experimenter web page for the derived objects and topics. The object detection team worked with Faster R-CNN and YOLOv7 models, and implemented post-processing rules for saving objects in a structured format. As the final deliverable for object detection, inference on 5k ETDs has been completed, and the refined objects have been saved to the Repository. The topic modeling team worked with clustering ETDs to 10, 25, 50, and 100 topics with different models (LDA, NeuralLDA, CTM, ProdLDA). As the final deliverable for topic modeling, we store the related topics and related documents for 5k ETDs in the Team 1 database, so that Team 2 could provide the related topic and documents on the documents page. By the end of the semester, the team had the deliverables being the Experimenter web page for the derived objects and topics, and the related objects and topics for 5k ETDs stored in the Team 1 database.

**Keywords:** ETD, electronic theses and dissertations, deep learning, object detection, topic modeling

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Overview

## 1.1 Introduction

In 1997, Virginia Tech was the first university to require its students to submit their theses and dissertations (ETDs) electronically. Recent research at Virginia Tech and Old Dominion University has collected over 500,000 ETDs from around the USA that provides our corpus. Accordingly, we are trying to make them more accessible for searching by extracting the metadata, chapters, abstracts, table of contents, figures, and so on as well as implementing topic modeling algorithms to obtain a list of the most relevant topics associated with the document files. Eventually, users will have access to the ETDs through the web pages developed by the Fall 2022 CS5604 class. In this report, we discuss the pipeline of our work as well as the architecture, design, and implementation of the underlying algorithms that we have employed.

There are 5 teams to achieve this goal. A curator team (Team 1) stores the ETDs and outputs from other teams and provides APIs to access this data. Another team (Team 2) for the front-end user persona allows users to search ETDs and use a recommendation system. Our team will experiment with topic models and object detection which will ultimately allow searching of ETDs through specific topics by a chapter and section level. Team 1 enables uploading all data contained in an ETD to the database. Another experimenter team (Team 4) would deal with analysis of ETDs through language models, classification, and summarization by a chapter and section level. All of the services would be provided by the DevOps team (Team 5) that would provide CI/CD services through Docker containers.

Our role in this project will be experimenting with topic models and object detection. The refined objects from the object detection pipeline along with the list of topics from the topic modeling pipeline will be fed into Team 1's Repository. Figure 1.1 shows the general pipeline that our Objects & Topics team has been following throughout the semester.

The object detection subteam will extract the ETD objects using YOLOv7 and Faster R-CNN (Detectron2), and work on post-processing logic to parse the text and image objects from the detection. Both the list of objects (un-ordered set) and the ETD hierarchy would be stored in separate data structures, which would be used to generate an XML file, and create RDF Triples to be stored in the Knowledge Graph. The outputs can then be accessed from the Repository by other teams for downstream tasks.

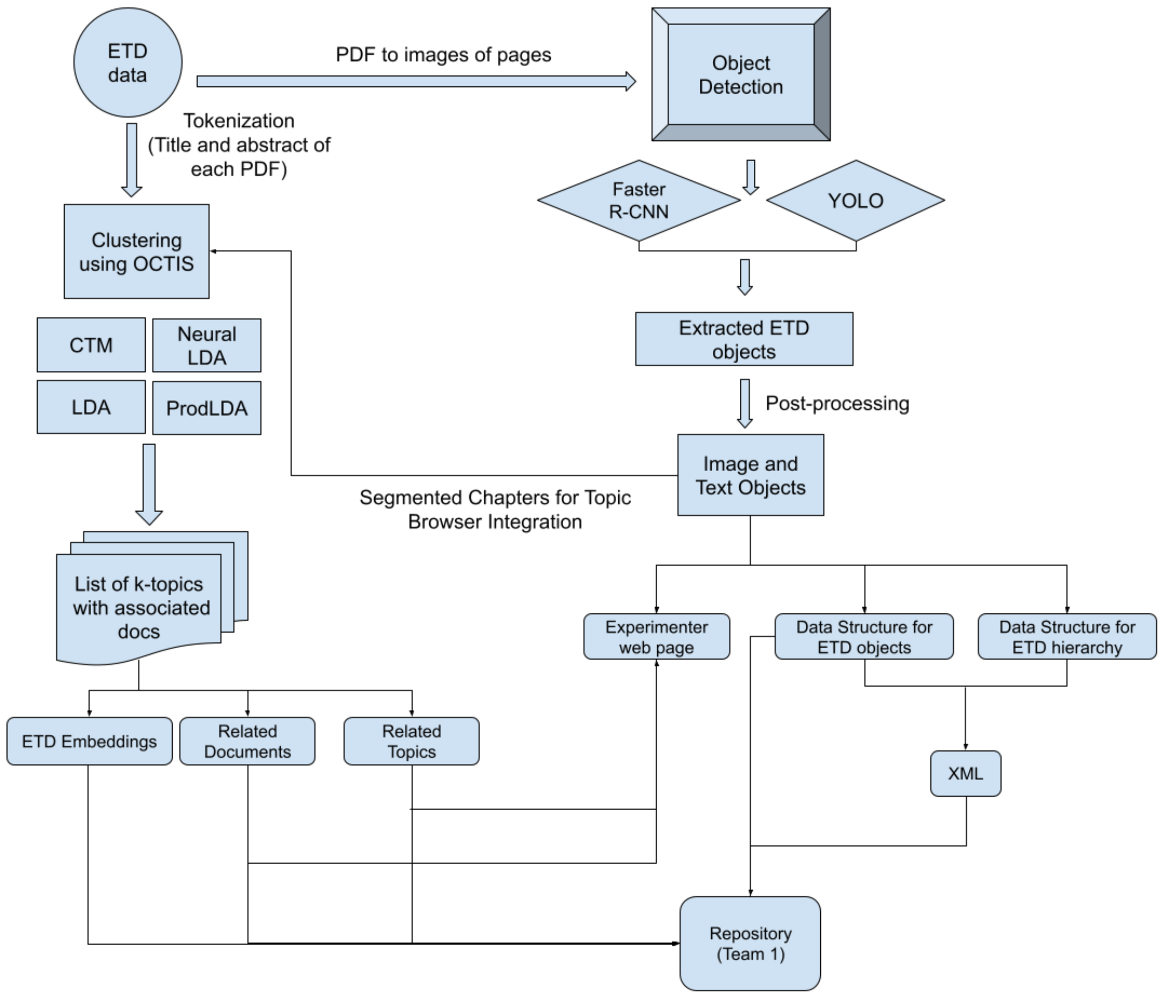


Figure 1.1: Complete pipeline diagram

The topic modeling team will train the models based on ETDs and segmented chapters generated by the object detection team, and work on topic modeling and clustering algorithms.

We use Team 1's API to store the ETD objects as well as the list of topics in the Repository. The complete pipeline will be explained in detail in the following sections.

## 1.2 Project Management

Actively managing human resources to get work done efficiently is one of the key requirements in this project. There are a host of project tracking tools like JIRA, Google Tasks, and Trello which can be clubbed with Git integration to track who is working on which feature and what all code changes has been done towards a particular task.

For our team communication, we decided to use Google Tasks owing to its simplicity. We keep track of the progress of this project through weekly meetings, where each of us updates the others with the current progress and challenges faced. Thus whenever some issue comes up, everyone can instantly resolve that issue together. Besides, our project client could assign tasks and set due dates, so the work flow is transparent and efficient.

# Chapter 2

# Literature Review

Research exploration of literature is a time consuming process. A researcher has to go through hundreds of documents, filter out the ones which are of interest and develop further analysis. This iteration has to go multiple times in most cases. Our class is collaborating with the University Library at Virginia Tech [1] to build an information storage and retrieval system to help speed up that process. They have given us a collection of over 500,000 ETDs [2] from different universities across the country. Each ETD has some metadata fields which will be described in the subsequent sections. Along with the data, we have the full text and the PDF version of the ETD document. At present, topic modeling and object detection models are helping literature reviewers by suggesting papers which can be more relevant to the reader.

## 2.1 Dataset

Our dataset has more than 500K ETDs ranging from 1845 to 2020, but most of them are published after 1945.

### 2.1.1 Object Detection

The labeling for the ETD objects was done using Roboflow [3], a free online labeling tool. There are 24 categories that were identified and labeled which were deemed common in ETDs such as paragraph, figure, and table. Table 2.1 contains all the categories that the detection model will identify. There are 20 text-based classes, and 4 image-based classes.

The labeling was provided to us from 6 undergraduate students working during the Spring 2022 semester [4]. The dataset contains both born digital and scanned ETDs. During the Summer 2022 semester, our Subject Matter Expert (SME) augmented the dataset to add to the final object count of ∼25K pages (∼100K objects) to get the dataset we are working with for this project.

Table 2.1: ETD Categories

| **Category** | **#Instances** |
|---|---|
| Title | 439 |
| Author | 404 |
| Date | 338 |
| University | 309 |
| Committee | 282 |
| Degree | 279 |
| Abstract Heading | 169 |
| Abstract Text | 183 |
| List of Contents Heading | 512 |
| List of Contents Text | 1059 |
| Chapter Title | 2211 |
| Section | 9337 |
| Paragraph | 30359 |
| Figure | 6359 |
| Figure Caption | 5722 |
| Table | 2654 |
| Table Caption | 2213 |
| Equation | 5092 |
| Equation Caption | 3051 |
| Algorithm | 96 |
| Footnote | 5722 |
| Page Number | 24543 |
| Reference Heading | 313 |
| Reference Text | 2088 |
| Total Objects | 99859 |
| Total Images | 25073 |

### 2.1.2 Topic Modeling

The dataset contains the basic information of ETDs, such as ETD ID, title, and abstract. We took the title and abstract as our analyzing target. We also segmented chapters generated by the object detection group as extra analyzing targets.

## 2.2 Object Detection

We plan on evaluating our object detection pipeline using two models – Faster R-CNN and YOLOv7, as each of them has a distinct advantage.

Faster R-CNN, which was originally published in 2015 [5], is the third iteration of the R-CNN architecture [6] published by Ross Girshick et al. A second version called Fast R-CNN [7] replaced R-CNN's selective search algorithm with the Region of Interest (RoI) Pooling that enabled sharing expensive computations and made the inference much quicker. Faster R-CNN introduced a Region Proposal Network (RPN) that can be trained end-to-end with the CNN to generate high quality proposals using the feature maps from the CNN. It is to be noted that the original Faster R-CNN implementation used a standard VGG-16 backbone architecture for the CNN, whereas in the framework that we are using, Facebook AI Research's Detectron2 [8], there are object detection models built on top of Faster R-CNN with advanced architectures that perform really well. Though Faster R-CNN remained the state-of-the-art for years, it has been replaced by models with better speed vs. accuracy trade-offs such as the newer iterations of YOLO models. YOLOv7 has the highest accuracy of 56.8 percent average precision [9] when trained on the MS COCO dataset [14] from scratch without using pre-trained weights, and it outperforms both transformer-based object detectors and convolution-based object detectors in real-time. However, it is to be noted that most of our work concerning ETDs is done offline; they are processed in batches. Thus, the inclusion of both YOLO and Faster R-CNN models in the Experimenter web page would be beneficial.

## 2.3 Topic Modeling

In recent years, there have been many advances in the fields of text mining and natural language processing (NLP). Topic modeling in the analysis of ETDs aims to extract thematic collections of words that could represent topics from a large corpus of text documents. Several topic modeling algorithms have been proposed over the years. Earlier works such as Latent Dirichlet Allocation (LDA) [10] were based on the idea of probabilistic formulation of topics. Recent works in the domain of topic modeling include neural topic models such as NeuralLDA [11], ProdLDA [11], and CTM [12]. The representations learned from topic models can be used for downstream tasks that rely on document representations, such as finding similar documents (document recommendation), finding similar topics, and analyzing the variation of topics over time.

## 2.4 Tools

### 2.4.1 PyTorch

PyTorch [13] is a widely used open-source machine learning framework originally developed by Meta AI. A majority of the deep learning projects that are in deployment now use PyTorch. It supports Python [21] natively while also having a C++ interface. PyTorch and TensorFlow are considered to be the state-of-the-art frameworks for deep learning currently, before which frameworks like Caffe [19] and Torch [22] used to be popular. A major advantage in using PyTorch is that it provides much better GPU acceleration as it relies on Tensor operations at the base level.

### 2.4.2 YOLOv7

YOLOv7 is a real-time object detector that has greatly advanced the computer vision and machine learning world, as it is considered to be the fastest, and most accurate object detector till date. For training the model on the COCO dataset [14], pretrained weights were not used in the official YOLOv7 paper [9] by Chien-Yao Wang et al.

YOLOv7 uses a convolutional neural network to predict bounding boxes and class probabilities considering the entire image at one step, unlike earlier object detection models which use regions of the image to localize objects and images with high probabilities. The architecture is as follows – image frames are featured through a backbone (deep neural network composed of many convolutional layers, and that is used to extract certain features) which is then combined and mixed in the neck (additional layers between backbone and the head, which are referred to as a copy of detectors to collect feature maps from different stages) and then it finds the bounding boxes.

### 2.4.3 Detectron2

Detectron2 [8] is a framework for object detection built on top of PyTorch, developed by Facebook AI Research (FAIR). It has support for a number of research projects in computer vision and production applications within Facebook. It is a re-write from 2018's Detectron, which was written using Caffe2, which is now a part of PyTorch [13]. Besides object detection, Detectron2 also has support for advanced deep learning tasks such as activity recognition, semantic/instance segmentation, and several other tasks. Detectron2 has support for training object detection models on custom datasets using pre-trained weights that are provided in the official repository. All models in Detectron2 are pre-trained on the COCO dataset [14]. For our experiments, we would be using the Faster R-CNN models [5] present in Detectron2.

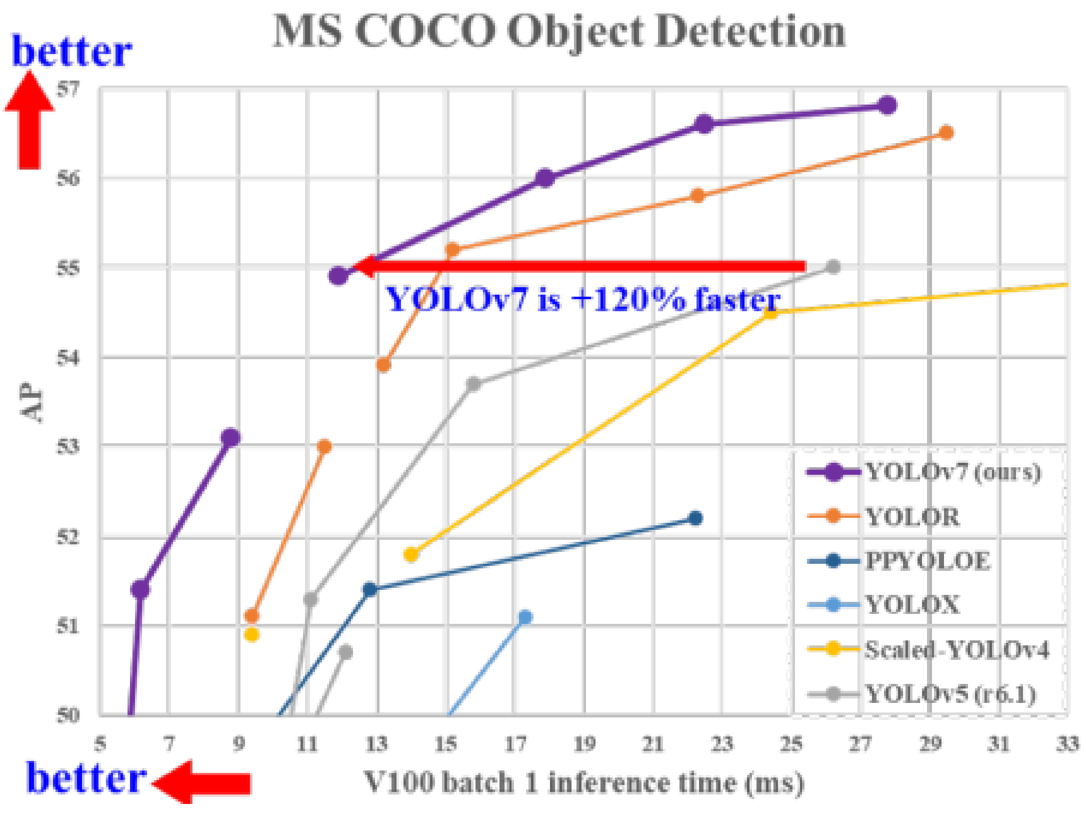


Figure 2.1: Comparison between real-time object detectors [9]

### 2.4.4 pdf2image

pdf2image [15] is a useful Python module that converts a Portable Document Format (PDF) file to PIL (Python Imaging Library) Image objects. The `convert_from_path` function returns a PIL Image object that represents each page of the PDF for which the local path is given as input. We implement this function as a pre-processing step to get the set of page images which would be given as input to the object detection model. Also, we would be storing the page images for ETDs in the Repository.

### 2.4.5 Flask

The Flask [16] framework is a lightweight web framework based on WSGI [17] implemented in Python, so it is convenient and straightforward for us to integrate it with our back end code. WSGI stands for Web Server Gateway Interface, and it describes how the web server communicates with the web application. Due to the reason that it is a micro-framework architecture, it is flexible and highly scalable, with a wide selection of third-party libraries. We can combine it with the Python libraries we need during the project development. In addition, the extension of new web features and functionalities would be easy to build. Besides, we also need the front-end technology, HTML, CSS, and JavaScript. HTML will provide the basic structure of the website, and CSS will control the styling of the website, while JavaScript will increase the interactivity with the users.

### 2.4.6 OCTIS

OCTIS (Optimizing and Comparing Topic models) [18] is a Python package that provides implementation of various topic modeling algorithms like Latent Dirichlet Allocation (LDA) [10], CTM [12], ProdLDA [11], and NeuralLDA [11]. The library also allows to evaluate the results of an algorithm by using classification metrics, coherence metrics, diversity metrics, and similarity metrics.

# Chapter 3

# Requirements

In this chapter, we discuss the requirements of our team: Objects & Topics.

## 3.1 Data Requirements

Our object detection model is a supervised learning algorithm which needs labeled data, i.e., each of the figures and tables within an ETD has to be explicitly listed for the training algorithms to learn. For the topic modeling algorithm, no labeling is required as it is an unsupervised algorithm. To satisfy our data requirements, Team 1 will be maintaining the Repository with Read/Write APIs using which our model will ingest raw data and give out processed output for catering to the end-users.

Although these were the proposed requirements, Team 1 wanted us to give them the model code for them to run and populate their own Repository. However, after discussing more with Team 1, we decided that it was better to store objects in the Repository using their POST APIs. We will discuss this in more detail in the following sections.

## 3.2 Processing Requirements

Both the object detection and topic modeling experiments are resource intensive. Due to the large volume of input data, we need GPUs (which are specialized hardware having several processors designed to perform one difficult task unlike a CPU designed to do several different tasks). A container on the cloud having the desired hardware will cater to our needs. For this, Team 5 has initialized a Docker instance with GPU enabled for our team. However, since the cloud server does not support Docker images larger than 20 GB in size, we had to scale down the requirements, and update the container to support the frameworks that we are using for both object detection and topic modeling.

# Chapter 4

# Design

## 4.1 Approach

As is shown in Figure 4.1, after creating the page images from the ETDs, we train YOLOv7 to extract objects such as the metadata, figures, tables, chapters, titles, and paragraphs where some are image-based and some are text-based objects. To extract text from the PDF images, we use OCR (for scanned PDFs) or PyMuPDF (for born digital PDFs). Finally, we pass the refined image and text objects to Team 1.

## 4.2 Tools

The tools/frameworks that are used in this projects are listed as follows:

- VT CS Cloud [23]
- GitLab [24]
- Docker [20]
- Python [21], PyTorch[13], Flask [16]
- Detectron2 [8] and YOLOv7 [9]
- LDA [10], Neural LDA [11], ProdLDA [11], CTM [12]
- OCTIS [18]

## 4.3 Methodology

### 4.3.1 Object Detection

From our initial assessment of the object detection models, we were able to conclude that YOLOv7 performs better than Faster R-CNN in terms of speed and accuracy for our use case,

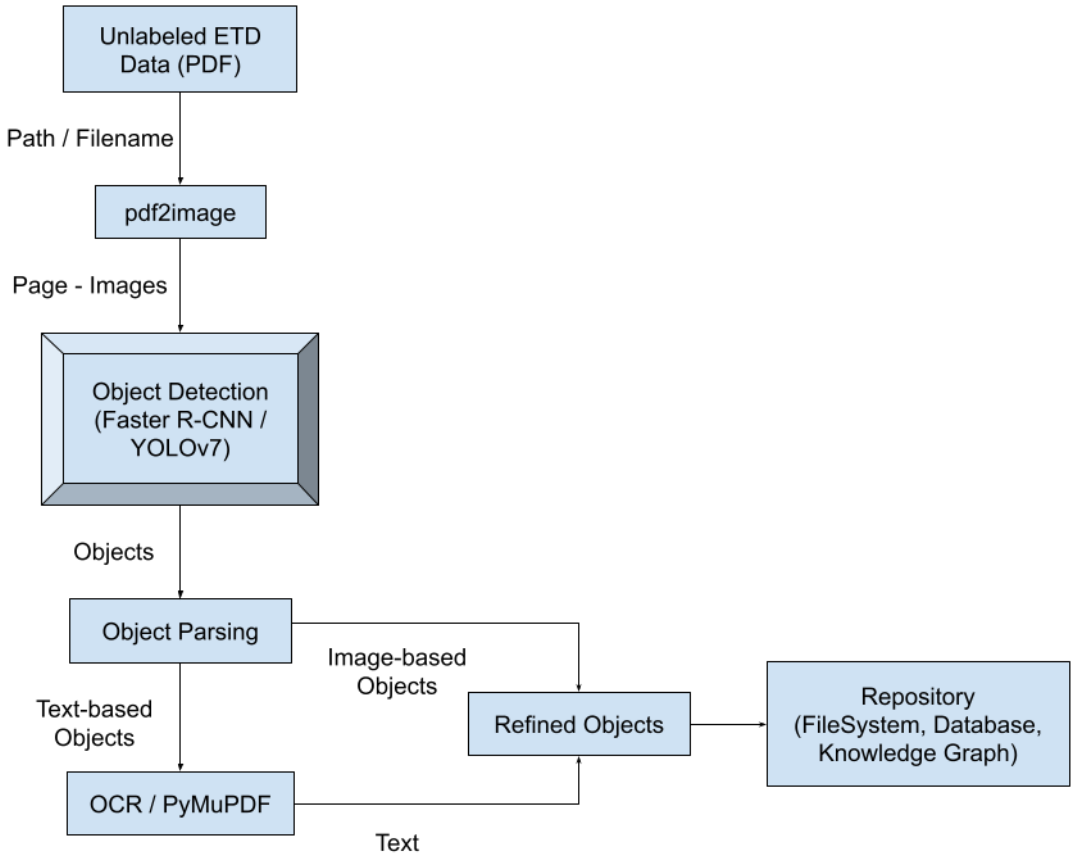


Figure 4.1: Object Detection – pipeline diagram

which is extracting different objects from ETDs. Therefore, we focus on the YOLOv7 model to implement our post-processing rules, though we give the experimenter the option to use both models (YOLOv7 and Faster R-CNN) as an additional feature with hyper-parameter tuning.

**Personas:** These include students and researchers who want to review work related to their research area, librarians and university administration who want an overview of recent research in their institutions, and experimenters working with object detection.

**Goals:** The object detection team is to provide an interface for the experimenter, supporting analysis of the results of an experimental instance of detection, for a specified set of ETDs. They will be able to run experiments with the Detectron2 and YOLOv7 models, with the option to input their own set of hyper-parameters to assess the accuracy of both the model and the post-processing rules.

For other types of personas, various services involving detected objects will be provided by other teams.

Next, we discuss the problems that came with the object detection models, and the solutions we implemented post-detection.

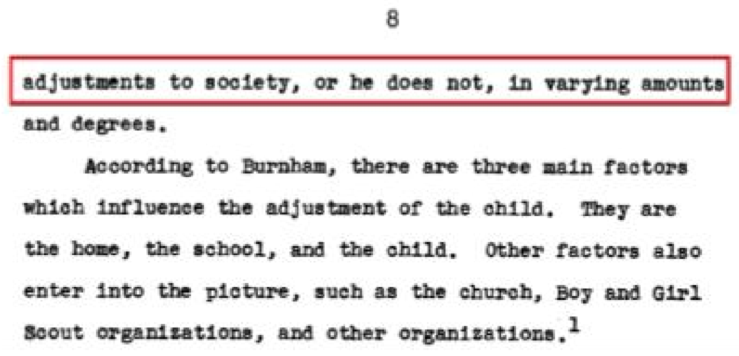
**False positives**

Problem:

- Last line of the paragraph in the previous page is detected as a chapter title as in Figure 4.2.

Solution (rule):

- Check the first letter of the title (capital).
- Avoid the text overflow for consecutive pages by checking the punctuation.
- Check if it is a subset of the table of contents.


8

adjustments to society, or he does not, in varying amounts and degrees.

According to Burnham, there are three main factors which influence the adjustment of the child. They are the home, the school, and the child. Other factors also enter into the picture, such as the church, Boy and Girl Scout organizations, and other organizations.[1]

Figure 4.2: Incorrectly detected chapter title – false positive

Problem:

- Image-based objects (figures, tables) and their corresponding captions might not be on the same page as in Figure 4.3.

Solution (rule):

- Determine the relationship between figure/table and their corresponding captions.

- Pair images with their corresponding captions according to the image being on top or bottom of images, or in some cases left or right.

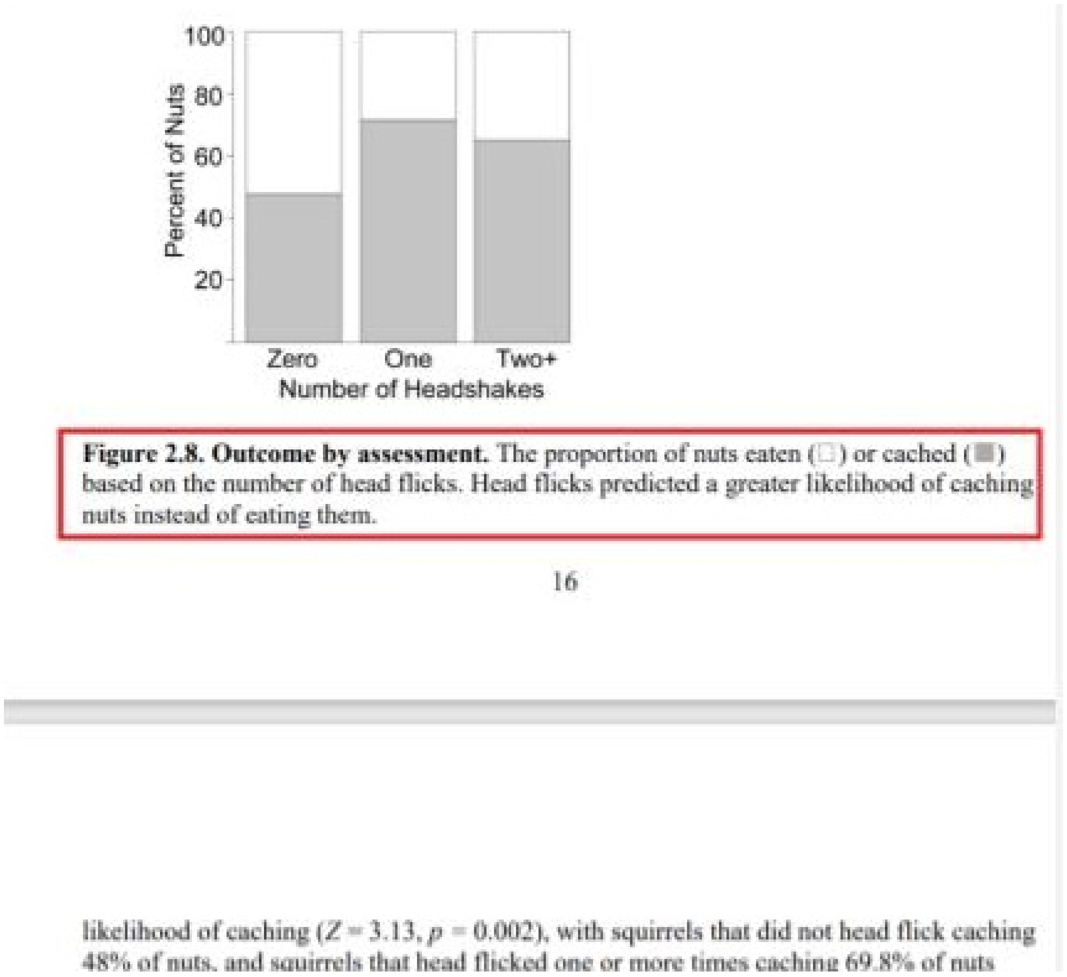


**Figure 2.8. Outcome by assessment.** The proportion of nuts eaten (□) or cached (■) based on the number of head flicks. Head flicks predicted a greater likelihood of caching nuts instead of eating them.

16

likelihood of caching ($Z = 3.13$, $p = 0.002$), with squirrels that did not head flick caching 48% of nuts, and squirrels that head flicked one or more times caching 69.8% of nuts

Figure 4.3: Incorrect caption detected – false positive

Problem:

- Chapter titles appear on multiple pages for the same chapter in some cases.

Solution (rule):

- Check if two consequent chapter titles are the same.

**ETD Filtering Rules**

The rules that we use to filter out incorrect detections, false positives, and map the hierarchy between objects are as listed below:

**Rules for filtering chapters/sections (delimiters)**

- The first letter of the detected text object should be capitalized.
- Whenever a new chapter tag is created (chapter title is detected), the last detected object from the previous page is checked. If it is a paragraph object, and the last character in paragraph is not ‘.’ then it is not considered to be a valid chapter tag.
- If a chapter title is the same as the previous chapter title, then a new chapter tag is not created.
- An empty section tag is created when a body object other than section name or chapter title is detected – this accounts for sections that do not have a chapter title or if the model fails to detect the chapter or section title.

**Rules for linking image-based objects with their captions**

- This rule has been implemented for two image-based objects – figure and table.
- The relationship between figure/table and their corresponding caption is first determined – captions can either appear on top or bottom of the image-based objects.
- The images are paired with their corresponding captions according to this pattern, assuming this pattern extends to the entire ETD.
- Exception to this rule – If the orientation of a page is is horizontal (landscape) then the pattern might not be consistent with the other pages. To avoid this, each page's orientation is checked, and if found to be horizontal, it is rotated by 90° – and a new set of detections (sorted by the y-axis) is obtained.

**Getting rid of false positives based on matching with table of contents (ToC)**

- Extract and save all the detected chapter titles, section titles, and page numbers using the text from the ToC.
- Create two files from comparison of ToC lines and chapter titles, one of which has the chapter titles similar to some lines of ToC, while the other has chapter titles that were not found in ToC. The same goes with section names and page numbers.
- Filter out the chapter and section titles that have been incorrectly detected.

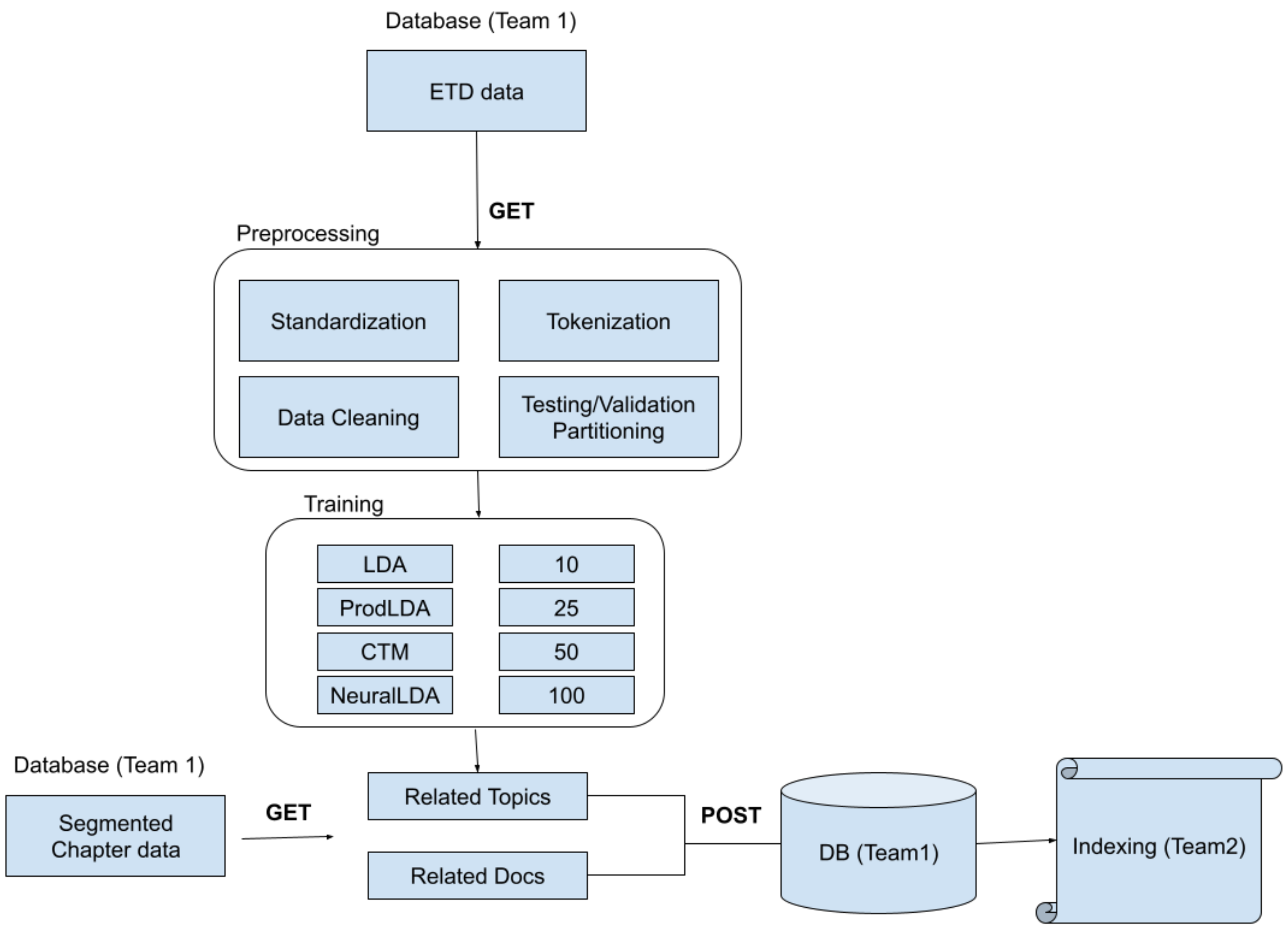


Figure 4.4: Topic Modeling – pipeline diagram

### 4.3.2 Topic Modeling

Topic Modeling involves clustering documents into list of topics using tokens and finding Nearest Neighbors using vectors to obtain most related documents. Earlier, the input comprised only of the ETD title and the overall abstract. But now, we incorporated the chapters and their summaries from the Team 1 database as input to our model. Figure 4.4 shows the complete data pipeline. As a result, our topic browser gives more fine-grained categories. They now link to a list of chapters and list of ETDs.

**Personas:** These include students and researchers who want to review work related to their research area, librarians and university administration who want an overview of recent research in their institutions, and experimenters working with topic modeling.

**Goals:**

1. Show the document topics based on different analysis models and numbers of desired topics.

   Right now, the services below are available to all the regular users who have the experimenter page link, but we will require login authorization in the future. The user will first select the model and the number of topics, which could refer to Figure 6.7. Next, the user will press the button for "select topic," which will lead the user to the topic page that displays several topics with an index (e.g., indication of related digital objects like ETDs and chapters) of the topic. After the user selects and presses the index of the topic, a page with documents that are topically related will be displayed.

2. Show related documents needed by users through topic models and indexes.

   The users first need to finish goal 1, then all relevant topics will be shown on a web page. Users can visit all topics and choose what they are interested in.

3. Show documents by a timeline search engine on specific topics. The figure example could refer to Figure 6.13.

   The user needs to first select the model and the number of topics. Next, the user will press the button for "select topic," which will lead the user to the topic page that displays several topics. After the user selects and presses the index of the topic, a page with documents that are topics related will be displayed. The user can enter the starting year, end year, and threshold on this page to search on the digital object(s) they want.

4. Show most relevant topics for a search query.

   The user inputs a set of keywords as a search query. Next, the user will select the model name and number of topics. After clicking the search button, a page with the top 10 results will be displayed where each result will indicate a topic (with associated hyperlink). On clicking any of these search results, the users will be redirected to show all of the documents connected with the selected topic.

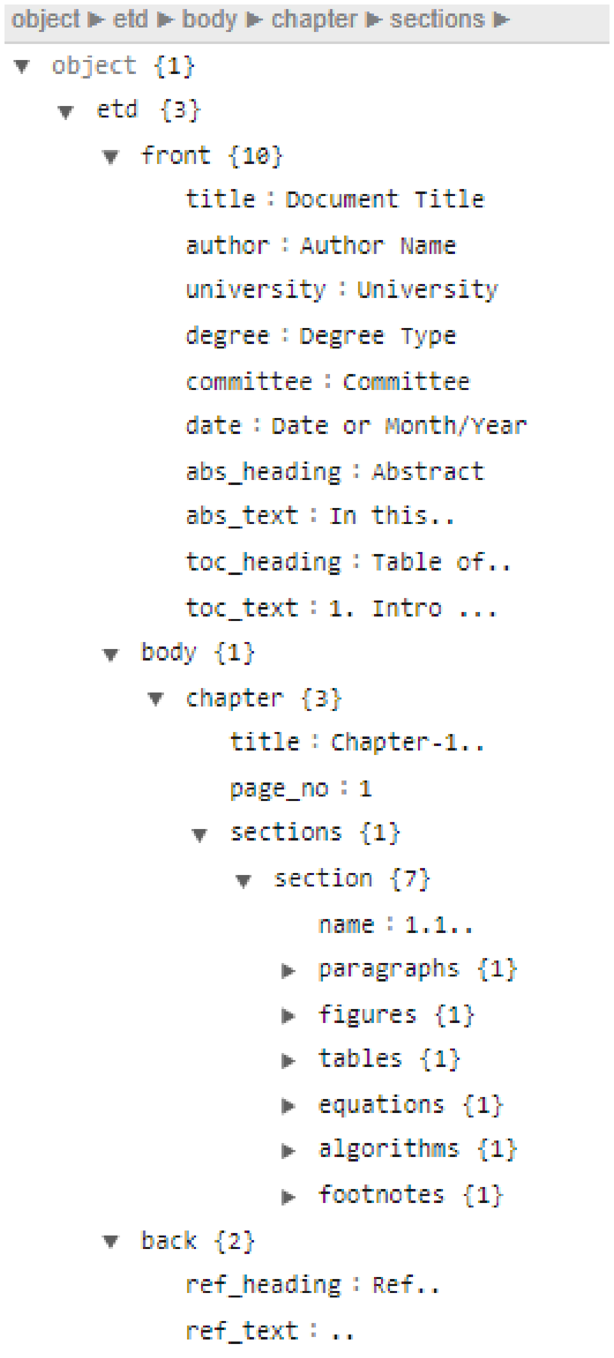
## 4.4 Deliverables

### 4.4.1 Object Detection

Our deliverables for object detection include:

- Extracted objects which will be stored inside Team 1's Repository.

- Experimenter web page which chooses either a Detectron2 (Faster R-CNN) or YOLOv7 model and displays all the extracted objects given an ETD.



```
object ▶ etd ▶ body ▶ chapter ▶ sections ▶
▼ object {1}
  ▼ etd {3}
    ▼ front {10}
        title : Document Title
        author : Author Name
        university : University
        degree : Degree Type
        committee : Committee
        date : Date or Month/Year
        abs_heading : Abstract
        abs_text : In this..
        toc_heading : Table of..
        toc_text : 1. Intro ...
    ▼ body {1}
      ▼ chapter {3}
          title : Chapter-1..
          page_no : 1
        ▼ sections {1}
          ▼ section {7}
              name : 1.1..
            ▶ paragraphs {1}
            ▶ figures {1}
            ▶ tables {1}
            ▶ equations {1}
            ▶ algorithms {1}
            ▶ footnotes {1}
    ▼ back {2}
        ref_heading : Ref..
        ref_text : ..
```

Figure 4.5: XML Schema

The XML schema to store the refined ETD objects is shown in Figure 4.5.

Each PDF has its ETD ID as the root element, and there are 3 sub-elements – front, body, and back. There are four image-based objects (figures, tables, equations, and algorithms) whereas the rest are text-based objects. The image-based objects have captions or numbers associated with them.

Table 4.1: Deliverables – ETD Objects

| **Output for a given ETD** | **Description/Contents** |
|---|---|
| Page images | Each page of ETD is saved as a JPG file. |
| Detected objects | Consists of sub-folders for each image-based object class (figure, table, equation, algorithm) saved as JPG files. |
| JSON object | Each object – text and image-based – is saved (unordered set as per the detection sequence). |
| Parsed XML | Detected objects in a tree structure with XML elements set to the clean text for text-based objects, and the image path for image-based objects. |

Based on our discussions with Team 1, and assessing the overall requirements of the class, we finalized the schema as shown in Table 4.1 to store our objects in the Repository.

An example JSON structure for a saved text-based object is in the following format:

```
{"type": "university",
"text": "University of California, Berkeley",
"page_num": 1, "bbox": [587, 1329, 1109, 1391]}
```

An example JSON structure for a saved image-based object is in the following format:

```
{"type": "figure",
"path": "1410_figure_98.jpg",
"page_num": 141, "bbox": [565, 170, 1140, 1887]}
```

The `type` key contains the class of the detected object (from a total of 24 classes as shown in Table 2.1). The `text` and `path` keys are set to the clean text if it is a text-based object or the saved image path if it is an image-based object, and the `bbox` key consists of the detected bounding box in this format: `[x1, y1, x2, y2]` where `(x1, y1)` represents the top-left corner, and `(x2, y2)` represents the bottom-right corner of the bounding box.

We are using Team 1's `Save detected object` (POST) API to store our objects.

`https://team-1-flask.discovery.cs.vt.edu/v1/objects/<etd_id>/<type_name>`

The `<type_name>` is set to `page`, `text`, `image`, and `xml`, to support different file types, and services required by the other teams.

Table 4.2: Deliverables – Objects API

| **Type** | **File** | **Metadata** |
|---|---|---|
| `page` | `<page_number>.jpg` | - |
| `text` | - | `{keys = type, text, page_num, bbox}` |
| `image` | `<etd_id>_<class>_<count>.jpg` | `{keys = type, path, page_num, bbox}` |
| `xml` | `<etd_id>.xml` | - |

The objects are stored in the Repository as shown in Table 4.2.

The deliverables for the other teams are listed as follows:

- Team 2: Experimenter web page
- Team 4: Refined objects (with hierarchy) for the chapter segmentation model
- Team 5: The requirements for them to set up the Docker containers for the cloud server, and provide services for workflow automation

We are providing Team 5 with 3 of our services (containerized), which are to be used for workflow automation, and our front-end web page. The services are as follows:

- Generate page images given an ETD ID
- Generate outputs given an ETD ID - using YOLOv7
- Generate outputs given an ETD ID - using Faster R-CNN (Detectron2)

These services are available through Docker images that generate the aforementioned outputs when an ETD ID that exists in the Repository is specified as input while executing. The first service splits a PDF into multiple pages and creates a sub-directory consisting of the individual pages saved as JPG files. The second and third services generate the required outputs (detected images, JSON objects, and the XML file) using YOLO and Faster R-CNN models, respectively. The detailed instructions on how to use these services are given in Section 7.

### 4.4.2 Topic Modeling

The deliverables for Topic Modeling include supporting APIs to return related topics and documents for an ETD, and a Topic Browser which visualizes all of the topics generated by the model and their respective documents. Table 4.3 lists the above mentioned points succinctly.

Table 4.3: Deliverables for Topic Modelling

| **Output** | **Description** |
|---|---|
| Related Topics & Related Docs | For a given ETD ID or Chapter ID, find the most similar topics and documents. |
| Topic Browser (Experimenter) | Allow a user to browse all topics, click on one and read associated documents (Documents by Topic, Chapters by Topic). |

#### APIs

We are currently offering 3 kinds of API calls. The ETD embedding API call takes in the model name, number of topics, and ETD ID, and returns a corresponding topic vector which is a vector containing the probability of the document belonging to each of the k different topics generated. The ETD related documents API takes in the model name, number of topics, and ETD ID, and returns the most related documents, while the parameter top k value is optional and defaults to 5 (related documents). The ETD related topics API takes in the model name, number of topics, and ETD ID, and returns the most related topics, while the parameter top k value is optional and defaults to 5 (related topics).

**Code Sample – get Topic vector**
**GET** /API/get_etd_embeddings/{model_name}/{num_topics}/{etd_id}
**Path Parameters:**

1. **model_name**: String
   Slug for name of model. Can be either of LDA, CTM, NeuralLDA, or ProdLDA.
2. **num_topics**: Integer
   Replace this with the number of topics to be generated. Has to be either of 10, 25, 50, or 100.
3. **etd_id**: Integer
   Provide the etd_id for which you need the results.

**Response (Valid Query):**

```
Status: 200
> {"status":"success","vector":[0.0003150326374452561,
0.20719221234321594,0.01724390685558319,0.11466679722070694,
0.00031506028608419,0.000315034732921049,0.00031509360997006297,
0.00031502716592513025,0.00031508112442679703,0.00031509919790551066,
........ ]}
```

**Response (Invalid Query):**

```
Status: 200
> {"status":"failure", "message":"Given etd_id not found!"}
```

**Code Sample – Get related Documents**
**GET** /API/get_etd_relateddocs/{model_name}/{num_topics}/{etd_id}/{topk}
**Path Parameters:**

1. **model_name**: String
   Slug for name of model. Can be either of LDA, CTM, NeuralLDA, or ProdLDA.
2. **num_topics**: Integer
   Replace this with the number of topics to be generated. Has to be either of 10, 25, 50, or 100.
3. **etd_id**: Integer
   Provide the etd_id for which you need the results.
4. **topk**: Integer
   Provide the top K value, but this path parameter is optional and its default is 5 without specification.

**Response (Valid Query):**

```
Status: 200
> {"status":"success","body":[218606, 63948, 321952, 91535, 241780]}
```

**Response (Invalid Query):**

```
Status: 200
> {"status":"failure", "message":"Given etd_id not found!"}
```

**Code Sample – Get related Topics**
**GET** /API/get_etd_relatedtopics/{model_name}/{num_topics}/{etd_id}/{topk}
**Path Parameters:**

1. **model_name**: String
   Slug for name of model. Can be either of LDA, CTM, NeuralLDA, or ProdLDA.
2. **num_topics**: Integer
   Replace this with the number of topics to be generated. Has to be either of 10, 25, 50, or 100.
3. **etd_id**: Integer
   Provide the etd_id for which you need the results.
4. **topk**: Integer
   Provide the top K value, but this path parameter is optional and its default is 5 without specification.

**Response (Valid Query):**

```
Status: 200
> {"status":"success","body":[11, 46, 70, 3, 22]}
```

**Response (Invalid Query):**

```
Status: 200
> {"status":"failure", "message":"Given etd_id not found!"}
```

**Services**

In order to support workflow automation (Team 5) we have divided our task into 2 services:

- Given a dataset having several ETDs, pre-process and train a Topic Model to generate a list of topics and their associated documents.
- For a given document (ETD/Chapter), return the top 5 most related topics and related documents.

These services will provide data to other teams as well as to our experimenter page, i.e., the Topic Browser.

# Chapter 5

# Implementation

## 5.1 Milestones

### 5.1.1 Object Detection

In Table 5.1 we list the tasks completed by the Object Detection team.

Table 5.1: Object Detection Milestones

| **Timeline** | **Tasks** |
|---|---|
| 1 Sept | Get familiar with Detectron2 and YOLOv7 |
| 8 Sept | Set up cloud server and get the model trained |
| 13 Sept | Set up object detection pipeline |
| 19 Sept | PDF to text using YOLOv7 |
| 23 Sept | PDF to images using YOLOv7 |
| 30 Sept | Parsing by chapters, sections and paragraphs |
| 4 Oct | Linking image based objects to caption / number |
| 6 Oct | API function to return a list of objects for an ETD |
| 15 Oct | Test parser logic and detect false positives |
| 15 Oct | Proposed rules method for removing false positives |
| 15 Oct | Complete object post-processing |
| 21 Oct | HTML generation for YOLOv7 from refined objects in a Flask app |
| 24 Oct | Team 1 deliverables – DB tasks (read/write) |
| 28 Oct | Added support for Detectron2 (Faster R-CNN) |
| 28 Oct | Finalized base models and parser logic for YOLOv7 and Faster R-CNN |
| 1 Nov | Integrated Detectron2 model with the Flask App |
| 4 Nov | Deployed the containerized Flask app on the cloud server |
| 28 Nov | Experiment with a bigger subset of the ETD dataset |
| Final | Run inference on 5k ETDs using Team 1's API to store into their DB |

### 5.1.2 Topic Modeling

In Table 5.2 we list out the tasks completed by the Topic Modeling team.

Table 5.2: Topic Modeling Milestones

| **Timeline** | **Tasks** |
|---|---|
| 1 Sept | Get familiar with LDA and OCTIS |
| 8 Sept | Run LDA from notebook on 500k dataset |
| 13 Sept | Create a table with etd_id and topic vector |
| 19 Sept | Set up Nearest Neighbour search |
| 23 Sept | Populate FaisNN with 230k ETD vectors from LDA model |
| 23 Sept | Perform similarity search (topic+document) through FaisNN |
| 30 Sept | Integrate FaisNN to Flask app |
| 3 Oct | Pagination of document results page |
| 9 Oct | Train on complete 500k ETD dataset and integrate with UI |
| 13 Oct | Improve topic filtering and query |
| 13 Oct | Display top-5 topics when topic search query is used |
| 19 Oct | Code refactoring and optimization for reducing load time |
| 26 Oct | API development and testing for populating Team 1's database |
| 28 Oct | Chapter dataset generation, pre-processing and training using XMLs from Object Detection |
| 6 Nov | Finalizing a Topic Browser front-end after trying multiple open source options with dummy data |
| 24 Nov | Inference learning of chapters from ETD based model |
| 27 Nov | Topic Bubble data generation complete. |
| 28 Nov | Pushing all chapter's related topics & docs to DB |
| 1 Dec | Push all ETD's related topics & docs to DB |
| Final | Set up workflow automation - Team 5 |

## 5.2 Developer Services and Workflow Automation

### 5.2.1 Object Detection

Since object detection inference is a resource intensive process, we have set up a GPU-enabled container [25] (with the required dependencies installed) on the cloud [23]. It is to be noted that support for both frameworks (YOLOv7 and Detectron2) has been included in the Docker Image used for this container. Also, we have provided the services for our object detection model (as explained in Section 4.4) packaged as separate Docker images to Team

5 for workflow automation, and for other teams to use these services, if required.

For Continuous Integration and Continuous Development (CI/CD), we are using the GitLab [27] instance on the cloud, maintained by Team 5. Currently, CI/CD features are only enabled for our Flask application that is running on the cloud. The application essentially converts the generated XML file to HTML, and renders it on the web page. Our front-end web page, with its functionality, is explained in detail in Section 7.

Workflow automation for the entire object detection pipeline has not been set up by Team 5. However, a basic workflow for the YOLOv7 service that we provided, has been set up, and is functional.

### 5.2.2 Topic Modeling

Since topic modeling training for a dataset is an expensive computation process, we did not set up a GPU-enabled container for that; that can be done as future work if sufficient resources are available for a suitable container. However, services that show our experimental pages have already been containerized by Team 5.

For Continuous Integration and Continuous Development(CI/CD), we are using the GitLab instance on the cloud, maintained by Team 5. These features have already been implemented by Team 5 in the GitLab repository. The application basically shows our experimental pages.

Workflow automation for the entire topic modeling pipeline has been not been set up by Team 5. However, a basic workflow for the Topic Browser application has been set up and is functional.

## 5.3 Inference

### 5.3.1 Object Detection

Due to time constraints, the original dataset of 50k ETDs was scaled down to 5k ETDs in order to align with the class goals. This collection of ETDs was given to us by Team 4, and the entire class was tasked with using this subset for their experiments. Since workflow automation was not set up to automate the process of running inference using multiple services, it was decided that we run the model on our containers, and use Team 1's API to store objects in the Repository. We are using a temporary directory on the container, that stores the inputs (using the GET API) and outputs (from the model inference), and finally the POST API is used to store the required files in the Repository. The file structure and outputs are as explained in Table 4.2. Inference on the 5k ETDs is completed, and the results are saved successfully to the Repository.

The `Get objects by ETD ID` API can be used to get the saved objects, and files, by specifying the `<etd_id>` and the `<type_name>`.

```
https://team-1-flask.discovery.cs.vt.edu/v1/etds/<etd_id>/objects?type=<type_name>
```

### 5.3.2 Topic Modeling

For a dataset size of 50k ETDs each having about 10 chapters, generating a model would have needed much more memory than is available to us during the course. To combat this, we have generated a model using just the 5k ETDs and performed inference for each chapter to generate respective data (related topics and documents).

Figure 5.1 illustrates the inference process we followed for each chapter. We iterated on each available chapter's raw text, performed preliminary data cleaning, and used the cleaned text as a query to perform similarity search on the model's topicwordmatrix. This way we get the top 5 related topics.

For related ETDs, we use the previously generated result of the top 5 topics to collect their respective top 5 ETDs. The overall list has 25 candidate ETDs which we sort based on their weights to get the top 5 related ETDs for the given chapter. This process is illustrated in Figure 5.2.

## 5.4 Challenges

### 5.4.1 Object Detection

A major challenge that we faced while parsing the extracted objects is false positives, as explained below. We implemented rules for the table of contents, chapters, sections, figures, and tables. One of the biggest challenges is that detection is not always in the top-bottom order for pages. This necessitates the creation of an XML tree since a hierarchy of the ETD would be required for other tasks, such as segmenting an ETD into its constituent chapters, and extracting image-based objects along with their corresponding captions. The XML tree structure also can be employed to promote accessibility, such as to provide a good order for a screen reader to follow when aiding those with low vision.

For various document parts and elements, the following lists some of the challenges, and

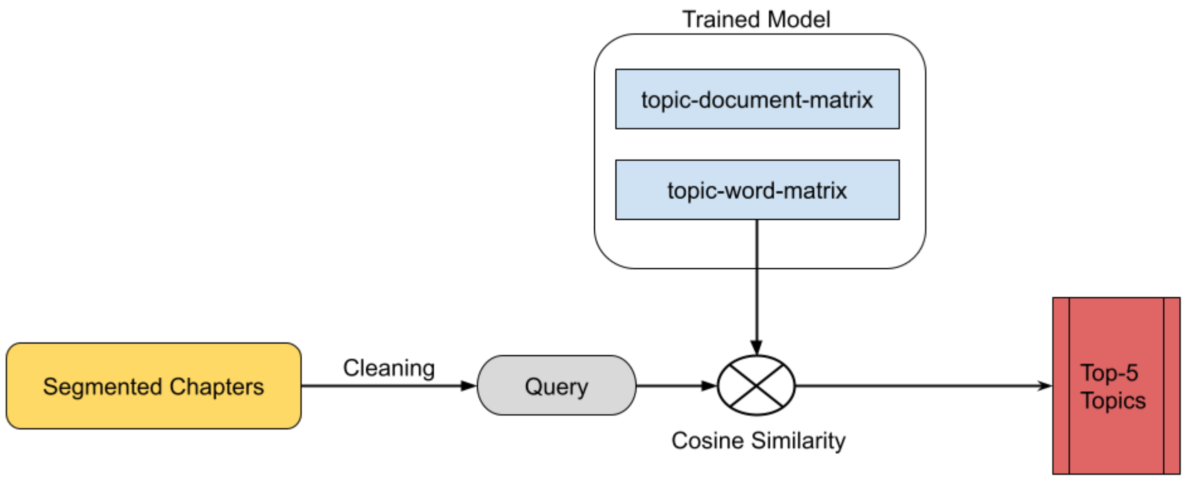


Figure 5.1: Related Topics using inference learning

rules to help with the analysis and XML tree construction.

- Front Metadata:
    1. Abstract is detected as title, and abstract text is detected as paragraph, and in some cases they overflow into the next page as paragraphs.
    2. Duplicate detections for some objects.
- Chapters and Sections (delimiters):
    1. Finding the right chapter title, since in some cases, chapter titles are being either not detected or wrongly detected.
    2. Refining chapter title detection by removing outliers.
    3. Finding if the keyword “chapter” is in the list of all detected chapter titles.
    4. Difference in font size between chapter titles and other objects.

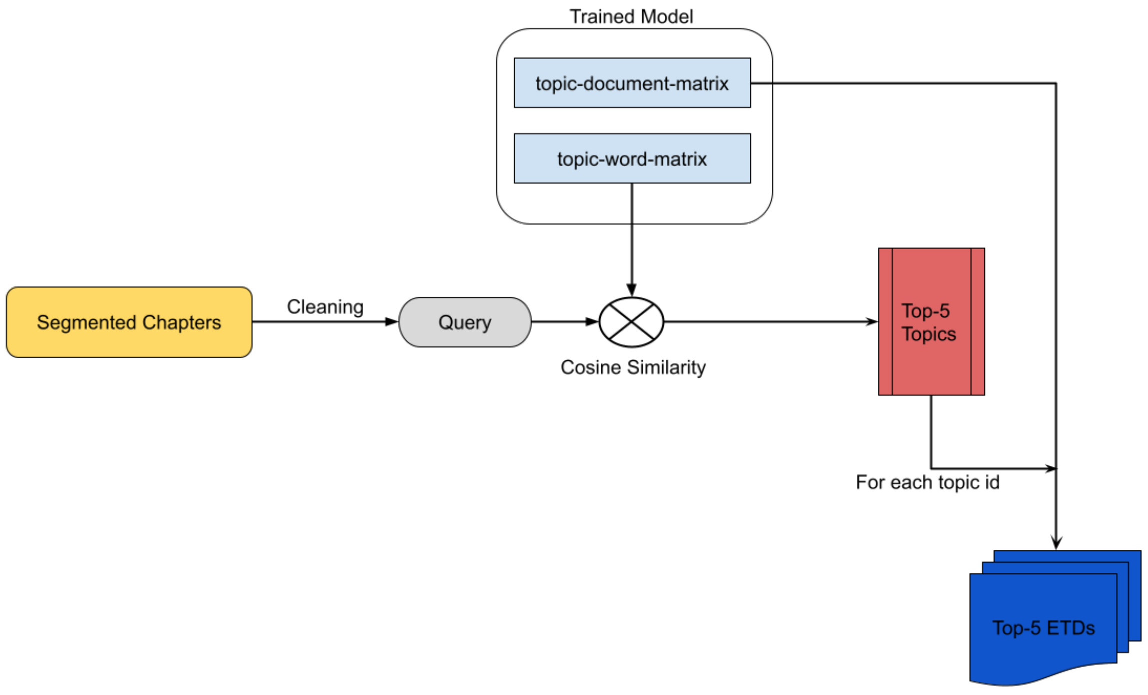


Figure 5.2: Related Documents using inference learning

5. Checking the indentation level – left/center alignment because chapter titles are either on the left side or at the center.
6. Chapter titles may appear on multiple pages as a header for the same chapter.

- Image-based Objects (algorithms, equations, tables, and figures):
  1. Linking the captions/numbers with their associated images based on Euclidean distance.
  2. Linking the captions/numbers with their associated images when they overflow into the next page.

### 5.4.2 Topic Modeling

The challenges faced in Topic Modeling overlap with the ones faced by the Object Detection sub-team. If the chapters generated from an ETD have errors due to the nature of input ETDs (like scanned images and random PDF layouts), then the output we get after using the erroneous data will also be incorrect.

Other than that, there also is a challenge to fit as much training data into the limited amount of RAM available from the cloud. Finally, we decided to reduce the dataset size to 5000 ETDs to solve it but that is leading to the degradation of cluster output.

## 5.5 Evaluation

### 5.5.1 Object Detection

We randomly sampled a small subset (around 50 ETDs) from the original dataset to experiment with our trained YOLOv7 and Faster R-CNN models, and tested both models with different hyper-parameter settings. Then, we refined the extracted set of ETD objects with our parser logic, and implemented the rules mentioned in Section 4.3.1 to filter out both incorrect detections and false positives. We finalized the set of rules for post-processing and extensively tested our parser logic with the object detection pipeline.

From running our model on the 5k ETDs, we observed that it takes around 1-2 minutes for each ETD to run our model on a container with GPU (in the cloud). If the model is run on a CPU without GPU, it takes about 5 minutes for each ETD. Before implementing the post-processing rules for our model, about 20% of the chapter titles and 5 percent of the captions were detected incorrectly, whereas after imposing the ETD filtering rules as specified in Section 4.3.1, we brought down the incorrect detections to about 3-4%.

### 5.5.2 Topic Modeling

We initially processed 2000 ETDs and trained the model in order to save time and increase efficiency. However, this also caused some unsatisfactory results in training, such as repeated topics and inaccurate topics while describing documents. Therefore, in the subsequent large-scale data processing, we were able to record diversity and coherence to ensure the acceptability of models, and we provided tables of diversity and coherence for users to browse as a tip to choose different models and set the number of topics according to their criteria. Below is the code snippet that shows how we determined the diversity and coherence.

```
# Initialize metrics
npmi=Coherence(texts=dataset.get_corpus(), topk=10, measure='c_npmi')
topic_diversity = TopicDiversity(topk=10)
# Retrieve metrics score
topic_diversity_score = topic_diversity.score(output)
print("Topic diversity: "+str(topic_diversity_score))
npmi_score = npmi.score(output)
```

```
        print("Coherence: "+str(npmi_score))
```

## 5.6 Future Work

In the interest of continuous improvement, we recommend future work to extend our efforts with both object detection and topic modeling. In this section we highlight the different plans of action post-class which could be implemented in the future.

### 5.6.1 Object Detection

The object detection group has mostly worked on the model, post-processing rules, and Experimenter UI. The biggest room for improvement was to be able to integrate better with the overall information system that the class has worked on. Here are a few highlighted downstream tasks that can be implemented to extend this project.

- Integrate the Flask app/Experimenter UI with the front-end and workflow services.
- Update our pipeline to support the `Add an ETD` service once Team 1 provides the API for saving metadata.
- Set up more workflow automation.
- Add support for scanned documents, and modify the existing pipeline accordingly.
- Add more post-processing rules based on the results of the 5k dataset.
- Improve UI layout, and add more functionality such as choosing hyper-parameters.
- Add test routines to the CI/CD pipeline.

### 5.6.2 Topic Modeling

In the future, it would be helpful to implement more models and integrate them into the topic browser. Besides that, the user interface can be further extended and improved. First and foremost would be to connect the Topic Bubble UI and our Topic Browser, so that users can switch between different applications. It also would help to further automate the workflow of the Topic Bubble application. Secondly, it would help to integrate our UI with Team 2 services, so that users who log in with accounts could access the experiment pages.

# Chapter 6

# User Manual

## 6.1 Object Detection

The front-end web page that the object detection team provides is the web page for experimenters. Figures 6.1 and 6.2 are the proposed wireframes that the team has come up with.

### 6.1.1 Proposed Wireframes

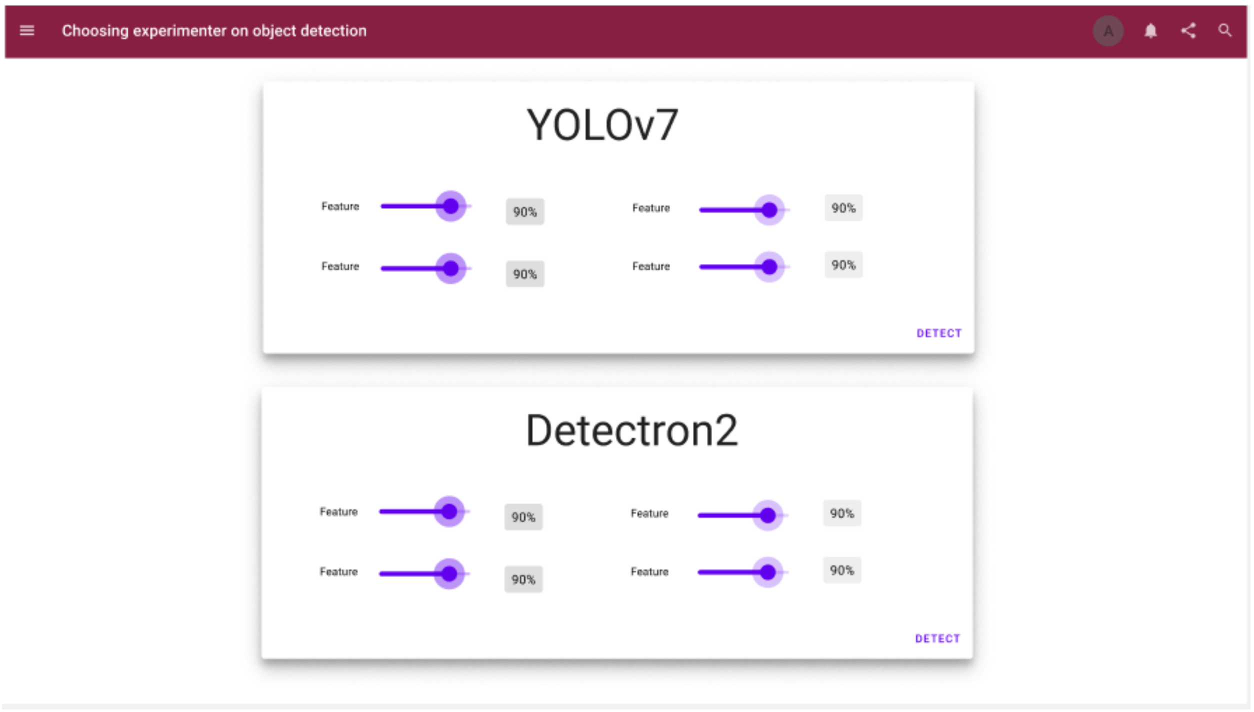


Figure 6.1: Object detection experimenter wireframe (Model selector page)

As is shown in Figure 6.2, the model selector page can support both the Detectron2 and YOLOv7 models. This also gives customization for different model weights and hyper-parameters. The Experimenter will be able to access this page by the sidebar menu, after logging in, and choosing object detection in the menu.

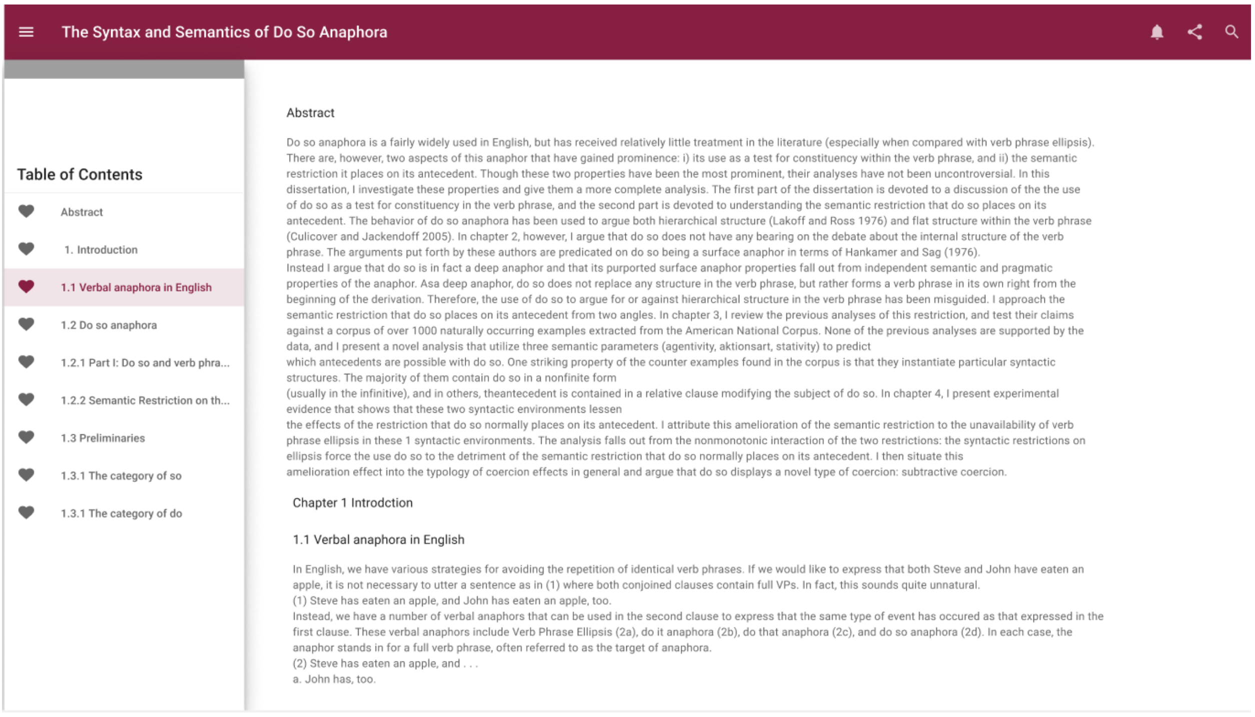

The Syntax and Semantics of Do So Anaphora

Table of Contents

- Abstract
- 1. Introduction
- 1.1 Verbal anaphora in English
- 1.2 Do so anaphora
- 1.2.1 Part I: Do so and verb phra...
- 1.2.2 Semantic Restriction on th...
- 1.3 Preliminaries
- 1.3.1 The category of so
- 1.3.1 The category of do

Abstract

Do so anaphora is a fairly widely used in English, but has received relatively little treatment in the literature (especially when compared with verb phrase ellipsis). There are, however, two aspects of this anaphor that have gained prominence: i) its use as a test for constituency within the verb phrase, and ii) the semantic restriction it places on its antecedent. Though these two properties have been the most prominent, their analyses have not been uncontroversial. In this dissertation, I investigate these properties and give them a more complete analysis. The first part of the dissertation is devoted to a discussion of the the use of do so as a test for constituency in the verb phrase, and the second part is devoted to understanding the semantic restriction that do so places on its antecedent. The behavior of do so anaphora has been used to argue both hierarchical structure (Lakoff and Ross 1976) and flat structure within the verb phrase (Culicover and Jackendoff 2005). In chapter 2, however, I argue that do so does not have any bearing on the debate about the internal structure of the verb phrase. The arguments put forth by these authors are predicated on do so being a surface anaphor in terms of Hankamer and Sag (1976).
Instead I argue that do so is in fact a deep anaphor and that its purported surface anaphor properties fall out from independent semantic and pragmatic properties of the anaphor. Asa deep anaphor, do so does not replace any structure in the verb phrase, but rather forms a verb phrase in its own right from the beginning of the derivation. Therefore, the use of do so to argue for or against hierarchical structure in the verb phrase has been misguided. I approach the semantic restriction that do so places on its antecedent from two angles. In chapter 3, I review the previous analyses of this restriction, and test their claims against a corpus of over 1000 naturally occurring examples extracted from the American National Corpus. None of the previous analyses are supported by the data, and I present a novel analysis that utilize three semantic parameters (agentivity, aktionsart, stativity) to predict
which antecedents are possible with do so. One striking property of the counter examples found in the corpus is that they instantiate particular syntactic structures. The majority of them contain do so in a nonfinite form
(usually in the infinitive), and in others, theantecedent is contained in a relative clause modifying the subject of do so. In chapter 4, I present experimental evidence that shows that these two syntactic environments lessen
the effects of the restriction that do so normally places on its antecedent. I attribute this amelioration of the semantic restriction to the unavailability of verb phrase ellipsis in these 1 syntactic environments. The analysis falls out from the nonmonotonic interaction of the two restrictions: the syntactic restrictions on ellipsis force the use do so to the detriment of the semantic restriction that do so normally places on its antecedent. I then situate this
amelioration effect into the typology of coercion effects in general and argue that do so displays a novel type of coercion: subtractive coercion.

Chapter 1 Introdction

1.1 Verbal anaphora in English

In English, we have various strategies for avoiding the repetition of identical verb phrases. If we would like to express that both Steve and John have eaten an apple, it is not necessary to utter a sentence as in (1) where both conjoined clauses contain full VPs. In fact, this sounds quite unnatural.
(1) Steve has eaten an apple, and John has eaten an apple, too.
Instead, we have a number of verbal anaphors that can be used in the second clause to express that the same type of event has occured as that expressed in the first clause. These verbal anaphors include Verb Phrase Ellipsis (2a), do it anaphora (2b), do that anaphora (2c), and do so anaphora (2d). In each case, the anaphor stands in for a full verb phrase, often referred to as the target of anaphora.
(2) Steve has eaten an apple, and . . .
a. John has, too.

Figure 6.2: Object detection parsed ETD wireframe (Extracted ETD view page)

After choosing a model, the experimenter will be redirected to the extracted ETD view page. As is shown in Figure 6.2, the generated XML from object detection will be used to generate an HTML page. This page will show all objects extracted from an ETD using the order of the XML schema, as discussed in Section 4.4.1.

### 6.1.2 Current Model

In order to visualize how the different models are producing the XML output on an ETD, we developed a Flask application that produces an HTML web page from the object detection model. In the future, we recommend that this be connected to the front-end website.

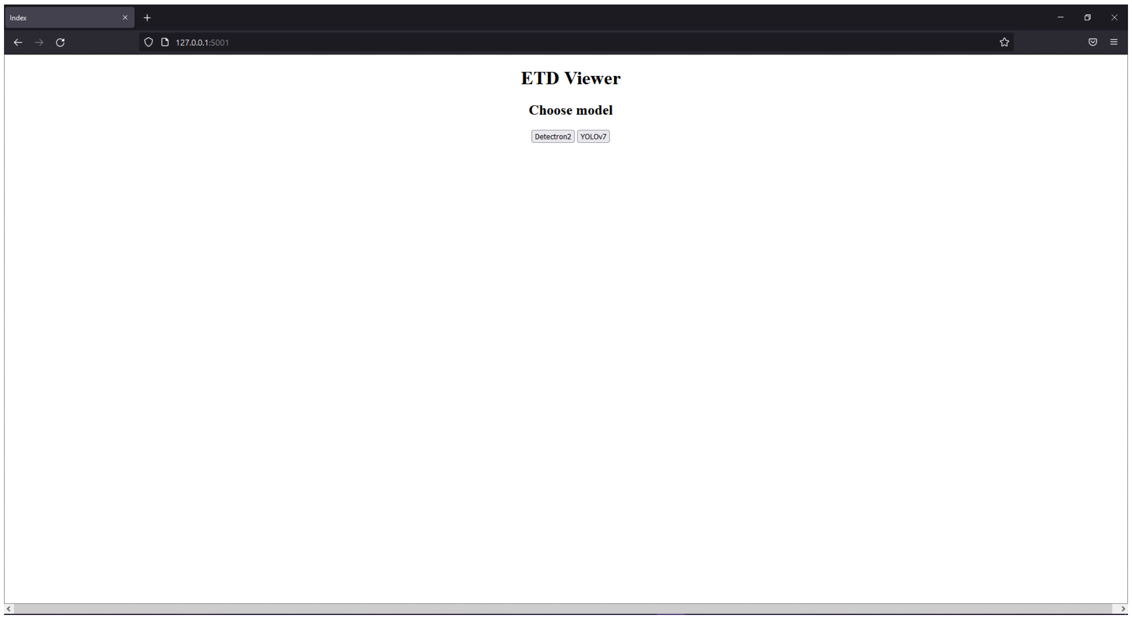


Figure 6.3: ETD viewer model selector page

In Figure 6.3, the user is given a choice to choose between the Detectron2 and YOLOv7 models for object detection.

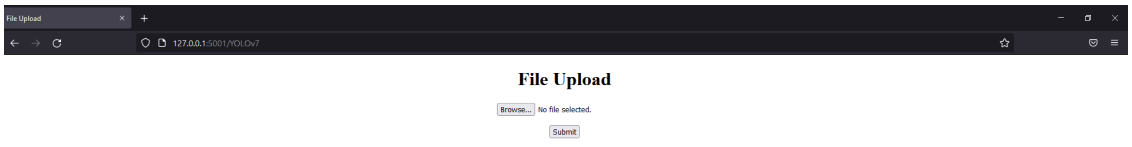
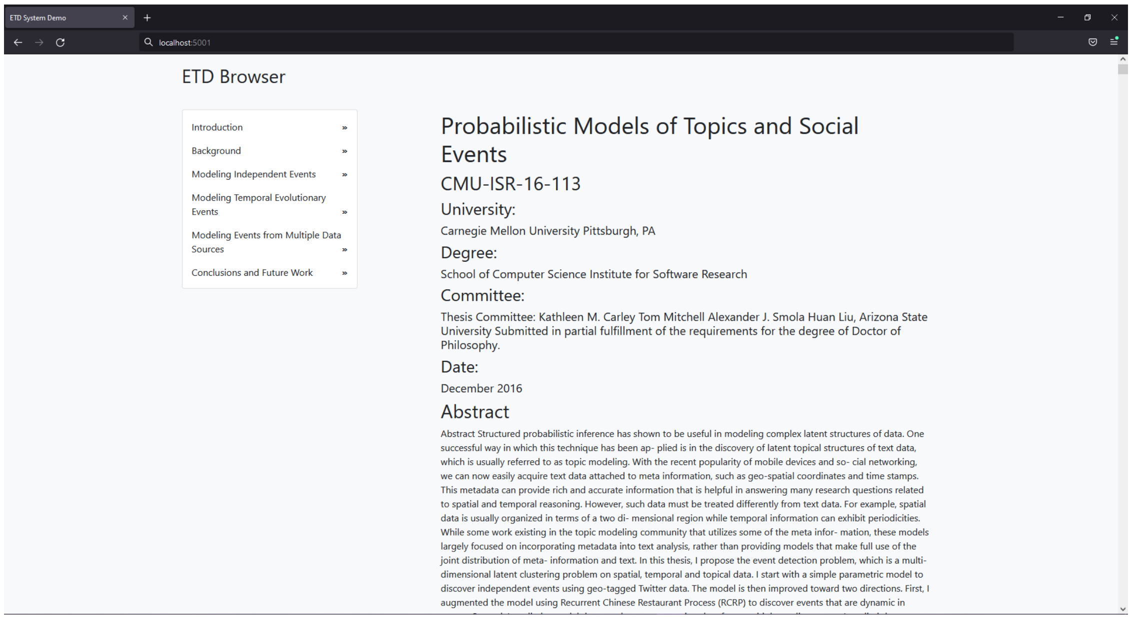


Figure 6.4: ETD viewer upload ETD page

After a user selects a model for object detection, they are given an option to upload a PDF of the ETD they want to perform the detection on. This is demonstrated in Figure 6.4.


ETD System Demo
localhost:5001
ETD Browser
Introduction »
Background »
Modeling Independent Events »
Modeling Temporal Evolutionary Events »
Modeling Events from Multiple Data Sources »
Conclusions and Future Work »

Probabilistic Models of Topics and Social Events

CMU-ISR-16-113

University:

Carnegie Mellon University Pittsburgh, PA

Degree:

School of Computer Science Institute for Software Research

Committee:

Thesis Committee: Kathleen M. Carley Tom Mitchell Alexander J. Smola Huan Liu, Arizona State University Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

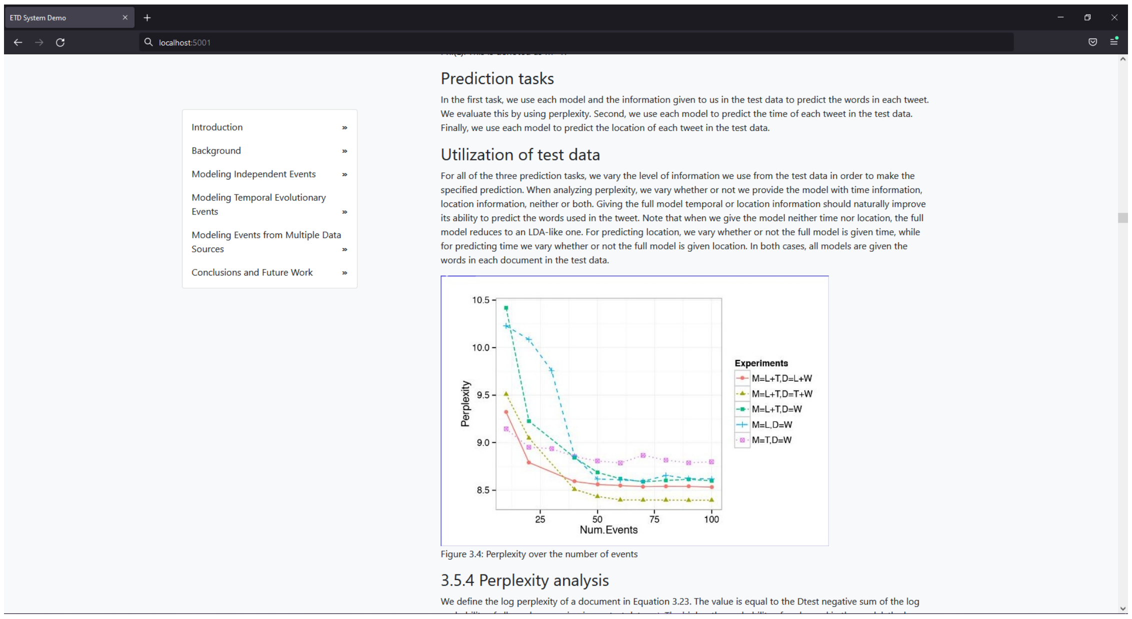
Date:

December 2016

Abstract

Abstract Structured probabilistic inference has shown to be useful in modeling complex latent structures of data. One successful way in which this technique has been ap- plied is in the discovery of latent topical structures of text data, which is usually referred to as topic modeling. With the recent popularity of mobile devices and so- cial networking, we can now easily acquire text data attached to meta information, such as geo-spatial coordinates and time stamps. This metadata can provide rich and accurate information that is helpful in answering many research questions related to spatial and temporal reasoning. However, such data must be treated differently from text data. For example, spatial data is usually organized in terms of a two di- mensional region while temporal information can exhibit periodicities. While some work existing in the topic modeling community that utilizes some of the meta infor- mation, these models largely focused on incorporating metadata into text analysis, rather than providing models that make full use of the joint distribution of meta- information and text. In this thesis, I propose the event detection problem, which is a multi-dimensional latent clustering problem on spatial, temporal and topical data. I start with a simple parametric model to discover independent events using geo-tagged Twitter data. The model is then improved toward two directions. First, I augmented the model using Recurrent Chinese Restaurant Process (RCRP) to discover events that are dynamic in

Figure 6.5: ETD viewer with metadata page



Prediction tasks

In the first task, we use each model and the information given to us in the test data to predict the words in each tweet. We evaluate this by using perplexity. Second, we use each model to predict the time of each tweet in the test data. Finally, we use each model to predict the location of each tweet in the test data.

Utilization of test data

For all of the three prediction tasks, we vary the level of information we use from the test data in order to make the specified prediction. When analyzing perplexity, we vary whether or not we provide the model with time information, location information, neither or both. Giving the full model temporal or location information should naturally improve its ability to predict the words used in the tweet. Note that when we give the model neither time nor location, the full model reduces to an LDA-like one. For predicting location, we vary whether or not the full model is given time, while for predicting time we vary whether or not the full model is given location. In both cases, all models are given the words in each document in the test data.

Figure 3.4: Perplexity over the number of events

3.5.4 Perplexity analysis

We define the log perplexity of a document in Equation 3.23. The value is equal to the Dtest negative sum of the log

Figure 6.6: ETD viewer with extracted image page

As is shown above, Figures 6.5 and 6.6 both show the produced HTML page that was generated from the object detection model. Figure 6.5 highlights the metadata fields while

Figure 6.6 shows the detected chapters and figures with linked captions extracted from the ETD. VTechWorks has two files associated with this report that illustrates the object detection results. One is the final class presentation. The other is an associated standalone video, ETDViewerDemo.mp4, regarding the ETD viewer. In Table 6.1, we give the timeline of events in ETDViewerDemo.mp4:

Table 6.1: Object Detection ETD viewer

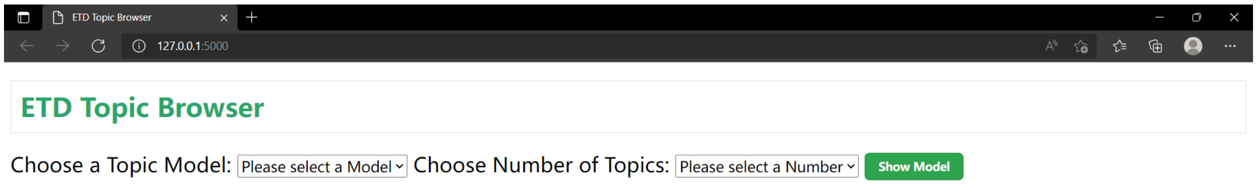
| **Time Stamp** | **Description** |
|---|---|
| 00:00 | Shows how one would upload an ETD to the application. |
| 00:10 | Loading ETD and running object detection. The user will be able to see the progress of the model. |
| 00:41 | Goes through the detected objects. The user is able to navigate to different chapters and sections of the ETD. |
| 00:43 | Demonstrates the functionality of the sidebar. |

## 6.2 Topic Modeling

The experimental front-end user interface (UI) encapsulates multiple downstream tasks and services for users of a digital library. Figure 6.7 shows the entry point for an end-user. Once the user selects a model and number of topics, the UI will lead the user to the topic list page. Figure 6.8 gives the topic list where a user could select their interest. It will lead to the documents page, shown in Figure 6.9, which gives an overview of ETDs related to that topic. Users can expand the abstract by clicking the 'show more button' as in Figure 6.10. Besides, users can click the title of the ETD and go to its documentation (see Figure 6.11) which has related topics and similar documents. The full text page in Figure 6.12 includes the full text URI button. The experimental system also supports filtering documents within a time range and university. That will lead the user to the result as shown in Figure 6.13. In order to provide a more intuitive and interactive UI, we made a Topic Modeling Bubble chart as shown in Figure 6.14, where each topic bubble represents a topic. The size and color of the topic bubble are determined by the number of documents related to that topic. Thus the larger and more red a topic bubble is, the higher is the number of related documents. After the user clicks a topic bubble, it will give a word cloud of the top 10 words in that topic, as seen in Figure 6.15. The topic bubble will expand to a larger rounded square when the user clicks on the blue expand button. That leads to the documents view and sources view as shown in Figure 6.16. The documents view gives the top 20 ETDs related to that topic, and shows the title and author information, while the sources view gives the author distribution. Clicking a word in the word cloud view, works as a filter to only show the topic bubble that contains the selected word. Meanwhile, it also presents the word weight in that topic as a pie chart as shown in Figure 6.17. In order to provide a more vivid presentation of the interactive UI, we made a video, TopicBubbleDemo.mp4. The key times and associated descriptions are shown in Table 6.2.

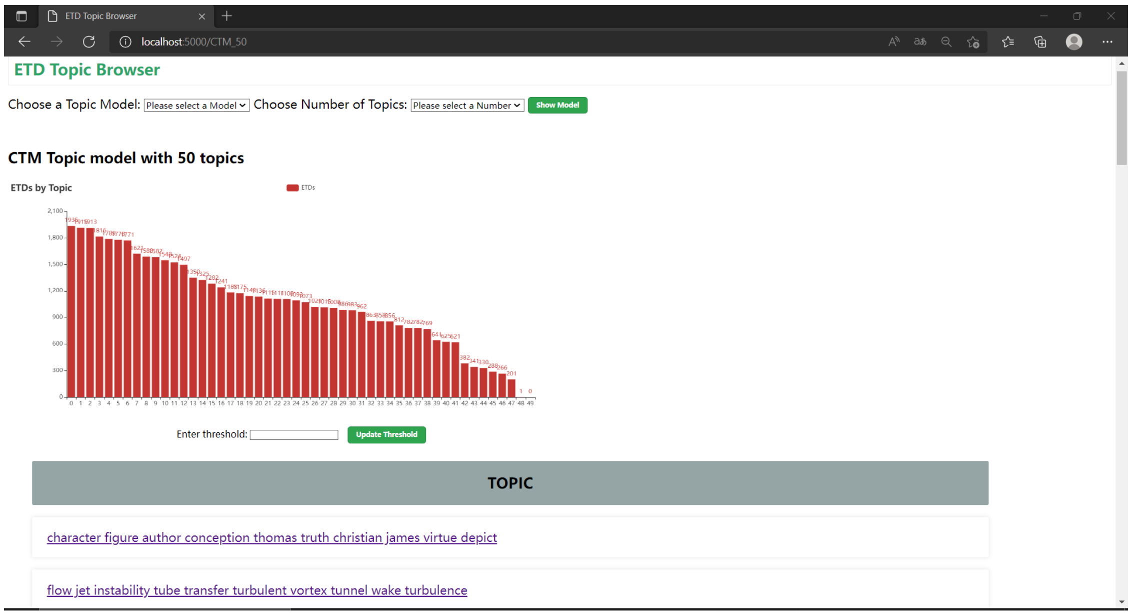
Table 6.2: Topic Modeling Topic Bubble UI

| **Time Stamp** | **Description** |
|---|---|
| 00:16 | Word cloud view after clicking the topic bubble |
| 00:23 | Documents view and sources view after clicking the blue expandable button |
| 00:34 | Filter words after clicking a word in the word cloud view |
| 00:51 | Locating the topic bubble by entering the topic number |


ETD Topic Browser

Choose a Topic Model: Please select a Model Choose Number of Topics: Please select a Number

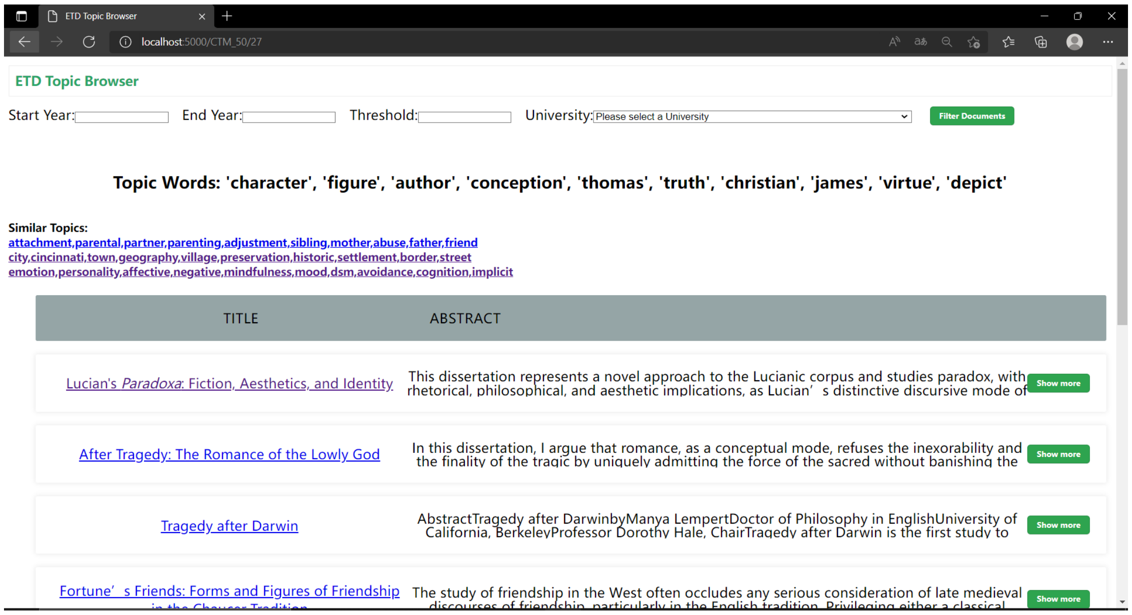
Figure 6.7: Index Page shows searching topic by selecting topic model and number of topics



character figure author conception thomas truth christian james virtue depict

flow jet instability tube transfer turbulent vortex tunnel wake turbulence

Figure 6.8: Topic Page shows browsing topic which is ranked by number of documents

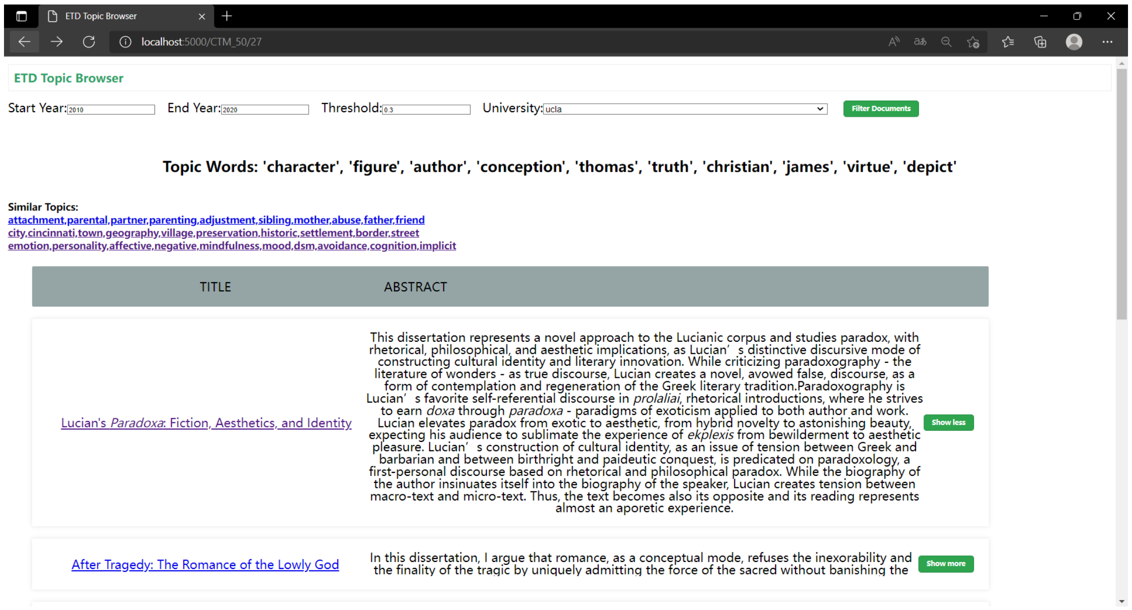


**Topic Words: 'character', 'figure', 'author', 'conception', 'thomas', 'truth', 'christian', 'james', 'virtue', 'depict'**

**Similar Topics:**
attachment,parental,partner,parenting,adjustment,sibling,mother,abuse,father,friend
city,cincinnati,town,geography,village,preservation,historic,settlement,border,street
emotion,personality,affective,negative,mindfulness,mood,dsm,avoidance,cognition,implicit

| TITLE | ABSTRACT |
|---|---|
| Lucian's *Paradoxa*: Fiction, Aesthetics, and Identity | This dissertation represents a novel approach to the Lucianic corpus and studies paradox, with rhetorical, philosophical, and aesthetic implications, as Lucian' s distinctive discursive mode of |
| After Tragedy: The Romance of the Lowly God | In this dissertation, I argue that romance, as a conceptual mode, refuses the inexorability and the finality of the tragic by uniquely admitting the force of the sacred without banishing the |
| Tragedy after Darwin | AbstractTragedy after DarwinbyManya LempertDoctor of Philosophy in EnglishUniversity of California, BerkeleyProfessor Dorothy Hale, ChairTragedy after Darwin is the first study to |
| Fortune' s Friends: Forms and Figures of Friendship in the Chaucer Tradition | The study of friendship in the West often occludes any serious consideration of late medieval discourses of friendship, particularly in the English tradition. Privileging either a classical |

Figure 6.9: Topic Selected Page shows browsing documents in a selected topic



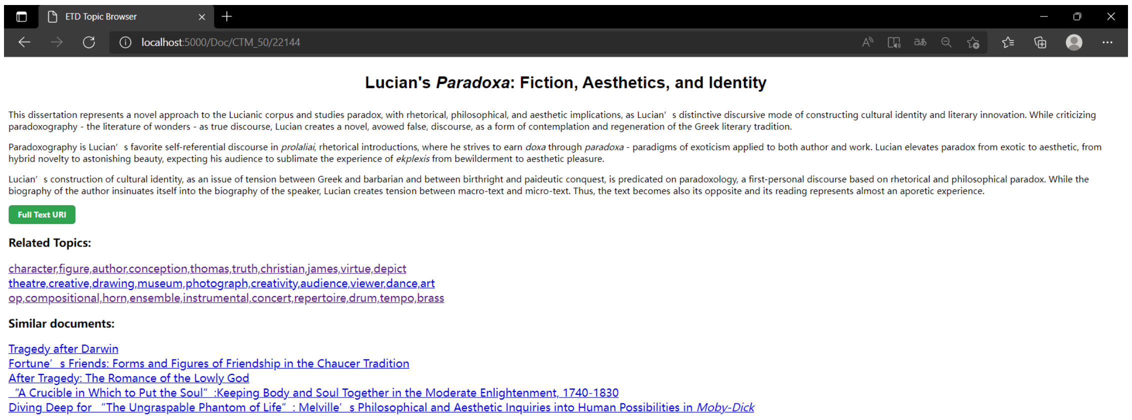
Start Year: 2010 End Year: 2020 Threshold: 0.3 University: ucla

**Topic Words: 'character', 'figure', 'author', 'conception', 'thomas', 'truth', 'christian', 'james', 'virtue', 'depict'**

**Similar Topics:**
attachment,parental,partner,parenting,adjustment,sibling,mother,abuse,father,friend
city,cincinnati,town,geography,village,preservation,historic,settlement,border,street
emotion,personality,affective,negative,mindfulness,mood,dsm,avoidance,cognition,implicit

| TITLE | ABSTRACT |
|---|---|
| Lucian's *Paradoxa*: Fiction, Aesthetics, and Identity | This dissertation represents a novel approach to the Lucianic corpus and studies paradox, with rhetorical, philosophical, and aesthetic implications, as Lucian' s distinctive discursive mode of constructing cultural identity and literary innovation. While criticizing paradoxography - the literature of wonders - as true discourse, Lucian creates a novel, avowed false, discourse, as a form of contemplation and regeneration of the Greek literary tradition.Paradoxography is Lucian' s favorite self-referential discourse in *prolaliai*, rhetorical introductions, where he strives to earn *doxa* through *paradoxa* - paradigms of exoticism applied to both author and work. Lucian elevates paradox from exotic to aesthetic, from hybrid novelty to astonishing beauty, expecting his audience to sublimate the experience of *ekplexis* from bewilderment to aesthetic pleasure. Lucian' s construction of cultural identity, as an issue of tension between Greek and barbarian and between birthright and paideutic conquest, is predicated on paradoxology, a first-personal discourse based on rhetorical and philosophical paradox. While the biography of the author insinuates itself into the biography of the speaker, Lucian creates tension between macro-text and micro-text. Thus, the text becomes also its opposite and its reading represents almost an aporetic experience. |
| After Tragedy: The Romance of the Lowly God | In this dissertation, I argue that romance, as a conceptual mode, refuses the inexorability and the finality of the tragic by uniquely admitting the force of the sacred without banishing the |

Figure 6.10: Show more Button Page shows full text of abstract



**Lucian's *Paradoxa*: Fiction, Aesthetics, and Identity**

This dissertation represents a novel approach to the Lucianic corpus and studies paradox, with rhetorical, philosophical, and aesthetic implications, as Lucian' s distinctive discursive mode of constructing cultural identity and literary innovation. While criticizing paradoxography - the literature of wonders - as true discourse, Lucian creates a novel, avowed false, discourse, as a form of contemplation and regeneration of the Greek literary tradition.

Paradoxography is Lucian' s favorite self-referential discourse in *prolaliai*, rhetorical introductions, where he strives to earn *doxa* through *paradoxa* - paradigms of exoticism applied to both author and work. Lucian elevates paradox from exotic to aesthetic, from hybrid novelty to astonishing beauty, expecting his audience to sublimate the experience of *ekplexis* from bewilderment to aesthetic pleasure.

Lucian' s construction of cultural identity, as an issue of tension between Greek and barbarian and between birthright and paideutic conquest, is predicated on paradoxology, a first-personal discourse based on rhetorical and philosophical paradox. While the biography of the author insinuates itself into the biography of the speaker, Lucian creates tension between macro-text and micro-text. Thus, the text becomes also its opposite and its reading represents almost an aporetic experience.

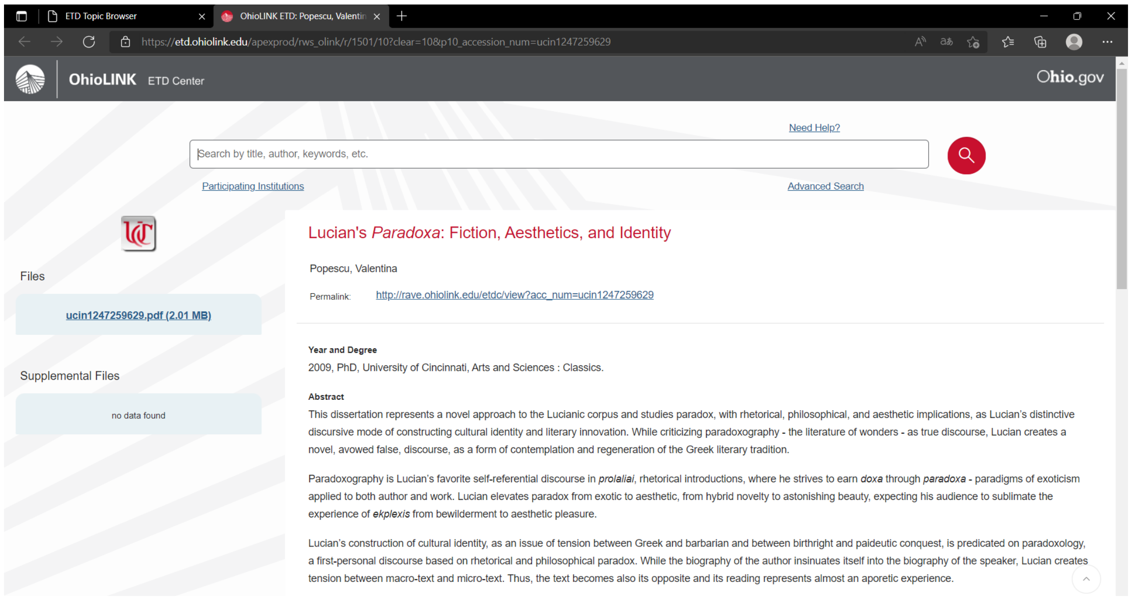
**Related Topics:**

character,figure,author,conception,thomas,truth,christian,james,virtue,depict
theatre,creative,drawing,museum,photograph,creativity,audience,viewer,dance,art
op,compositional,horn,ensemble,instrumental,concert,repertoire,drum,tempo,brass

**Similar documents:**

Tragedy after Darwin
Fortune' s Friends: Forms and Figures of Friendship in the Chaucer Tradition
After Tragedy: The Romance of the Lowly God
"A Crucible in Which to Put the Soul" :Keeping Body and Soul Together in the Moderate Enlightenment, 1740-1830
Diving Deep for "The Ungraspable Phantom of Life" : Melville' s Philosophical and Aesthetic Inquiries into Human Possibilities in *Moby-Dick*

Figure 6.11: Document Page shows specific information about the selected document



ETD Topic Browser | OhioLINK ETD: Popescu, Valentin

https://etd.ohiolink.edu/apexprod/rws_olink/r/1501/10?clear=10&p10_accession_num=ucin1247259629

OhioLINK ETD Center | Ohio.gov

Need Help?

Search by title, author, keywords, etc.

Participating Institutions | Advanced Search

Files

ucin1247259629.pdf (2.01 MB)

Supplemental Files

no data found

Lucian's *Paradoxa*: Fiction, Aesthetics, and Identity

Popescu, Valentina

Permalink: http://rave.ohiolink.edu/etdc/view?acc_num=ucin1247259629

**Year and Degree**

2009, PhD, University of Cincinnati, Arts and Sciences : Classics.

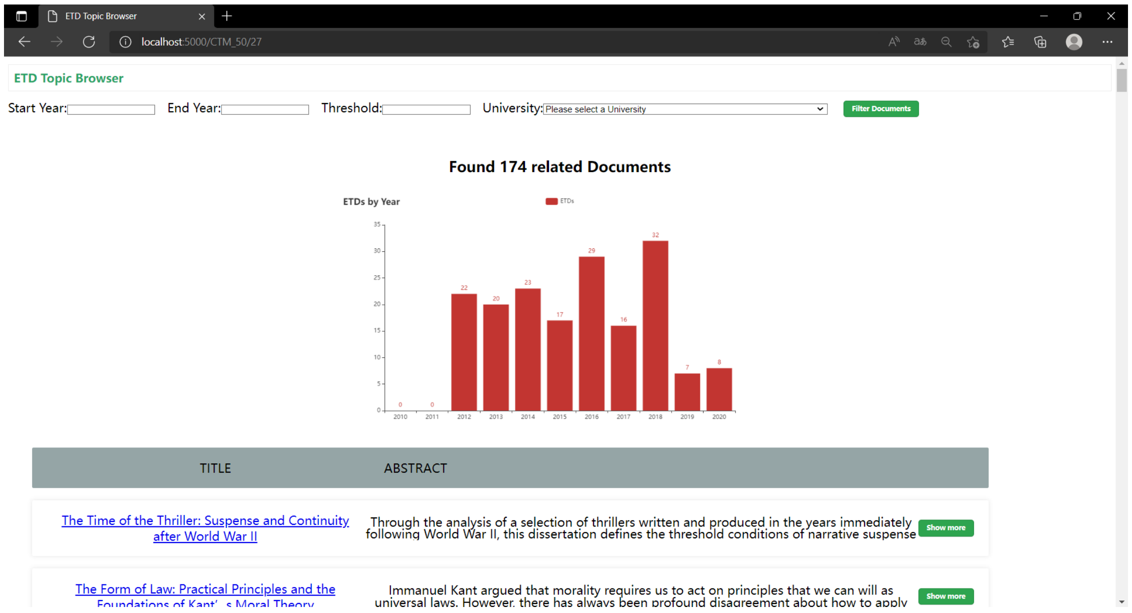
**Abstract**

This dissertation represents a novel approach to the Lucianic corpus and studies paradox, with rhetorical, philosophical, and aesthetic implications, as Lucian's distinctive discursive mode of constructing cultural identity and literary innovation. While criticizing paradoxography - the literature of wonders - as true discourse, Lucian creates a novel, avowed false, discourse, as a form of contemplation and regeneration of the Greek literary tradition.

Paradoxography is Lucian's favorite self-referential discourse in *prolaliai*, rhetorical introductions, where he strives to earn *doxa* through *paradoxa* - paradigms of exoticism applied to both author and work. Lucian elevates paradox from exotic to aesthetic, from hybrid novelty to astonishing beauty, expecting his audience to sublimate the experience of *ekplexis* from bewilderment to aesthetic pleasure.

Lucian's construction of cultural identity, as an issue of tension between Greek and barbarian and between birthright and paideutic conquest, is predicated on paradoxology, a first-personal discourse based on rhetorical and philosophical paradox. While the biography of the author insinuates itself into the biography of the speaker, Lucian creates tension between macro-text and micro-text. Thus, the text becomes also its opposite and its reading represents almost an aporetic experience.

Figure 6.12: Full Text URI Page shows full text source of the selected document



ETD Topic Browser

localhost:5000/CTM_50/27

**ETD Topic Browser**

Start Year: End Year: Threshold: University: Please select a University | Filter Documents

**Found 174 related Documents**

| TITLE | ABSTRACT |
|---|---|
| The Time of the Thriller: Suspense and Continuity after World War II | Through the analysis of a selection of thrillers written and produced in the years immediately following World War II, this dissertation defines the threshold conditions of narrative suspense (Show more) |
| The Form of Law: Practical Principles and the Foundations of Kant' s Moral Theory | Immanuel Kant argued that morality requires us to act on principles that we can will as universal laws. However, there has always been profound disagreement about how to apply (Show more) |

Figure 6.13: Filter Document Page shows filtering documents by start year, end year, threshold, and university

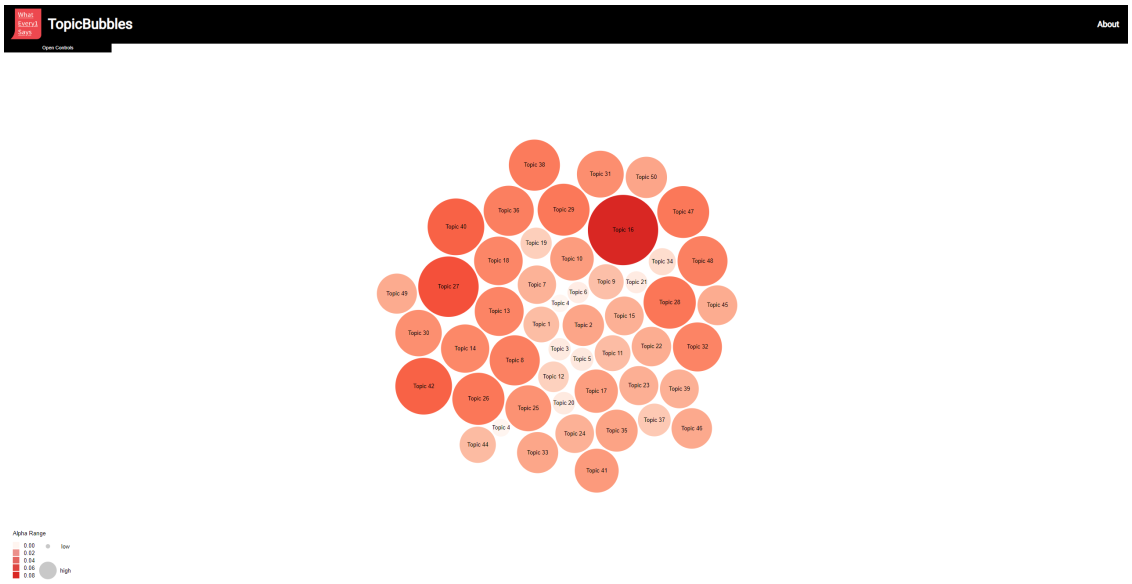


Figure 6.14: Topic Bubble page shows topic bubble where each bubble is represented as a topic

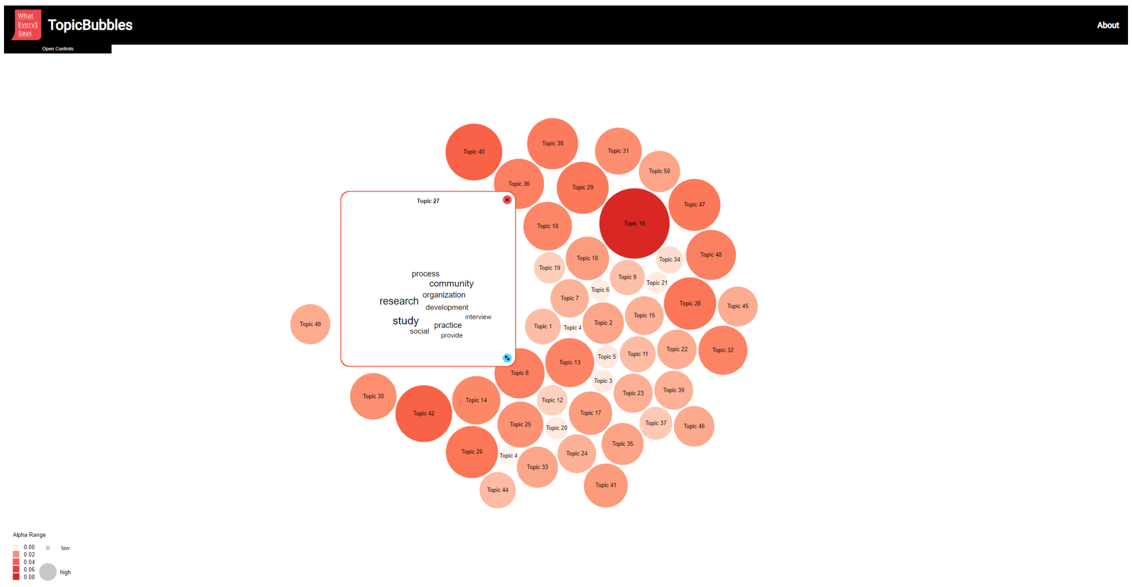
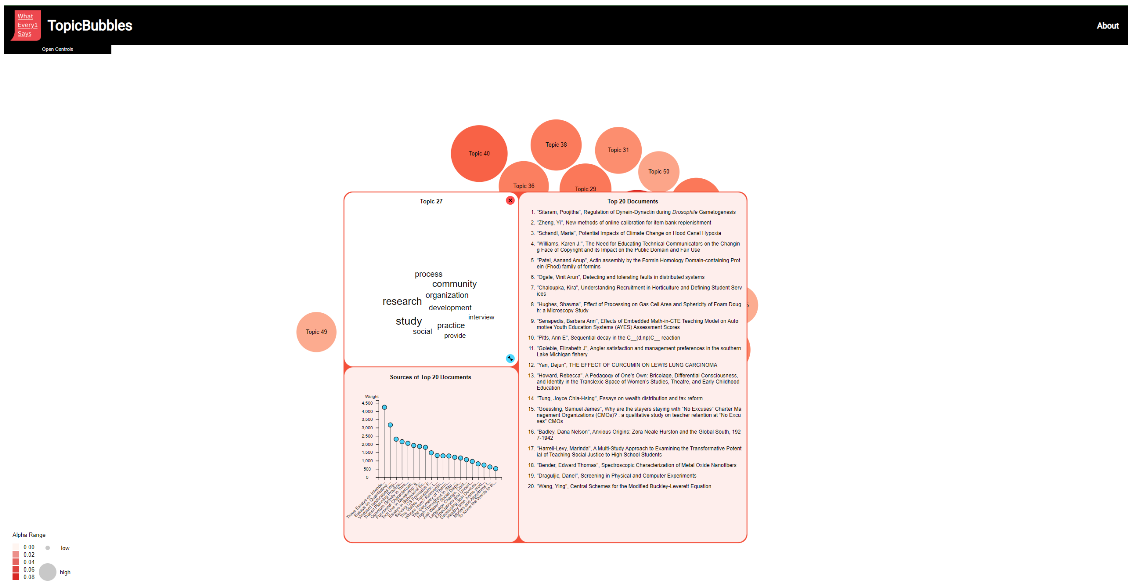


Figure 6.15: Topic bubble gives the word cloud view of the top 10 words when selecting a topic bubble



**Top 20 Documents**

1. "Sitaram, Poojitha", Regulation of Dynein-Dynactin during *Drosophila* Gametogenesis
2. "Zheng, Yi", New methods of online calibration for item bank replenishment
3. "Schandl, Maria", Potential Impacts of Climate Change on Hood Canal Hypoxia
4. "Williams, Karen J.", The Need for Educating Technical Communicators on the Changin g Face of Copyright and its Impact on the Public Domain and Fair Use
5. "Patel, Aanand Anup", Actin assembly by the Formin Homology Domain-containing Prot ein (Fhod) family of formins
6. "Ogale, Vinit Arun", Detecting and tolerating faults in distributed systems
7. "Chaloupka, Kira", Understanding Recruitment in Horticulture and Defining Student Serv ices
8. "Hughes, Shawna", Effect of Processing on Gas Cell Area and Sphericity of Foam Doug h: a Microscopy Study
9. "Senapedis, Barbara Ann", Effects of Embedded Math-in-CTE Teaching Model on Auto motive Youth Education Systems (AYES) Assessment Scores
10. "Pitts, Ann E", Sequential decay in the C__(d,np)C__ reaction
11. "Golebie, Elizabeth J", Angler satisfaction and management preferences in the southern Lake Michigan fishery
12. "Yan, Dejun", THE EFFECT OF CURCUMIN ON LEWIS LUNG CARCINOMA
13. "Howard, Rebecca", A Pedagogy of One's Own: Bricolage, Differential Consciousness, and Identity in the Translexic Space of Women's Studies, Theatre, and Early Childhood Education
14. "Tung, Joyce Chia-Hsing", Essays on wealth distribution and tax reform
15. "Goessling, Samuel James", Why are the stayers staying with "No Excuses" Charter Ma nagement Organizations (CMOs)? : a qualitative study on teacher retention at "No Excu ses" CMOs
16. "Badley, Dana Nelson", Anxious Origins: Zora Neale Hurston and the Global South, 192 7-1942
17. "Harrell-Levy, Marinda", A Multi-Study Approach to Examining the Transformative Potent ial of Teaching Social Justice to High School Students
18. "Bender, Edward Thomas", Spectroscopic Characterization of Metal Oxide Nanofibers
19. "Draguljic, Danel", Screening in Physical and Computer Experiments
20. "Wang, Ying", Central Schemes for the Modified Buckley-Leverett Equation

Figure 6.16: Topic bubble gives the documents view and sources view after clicking the expand button

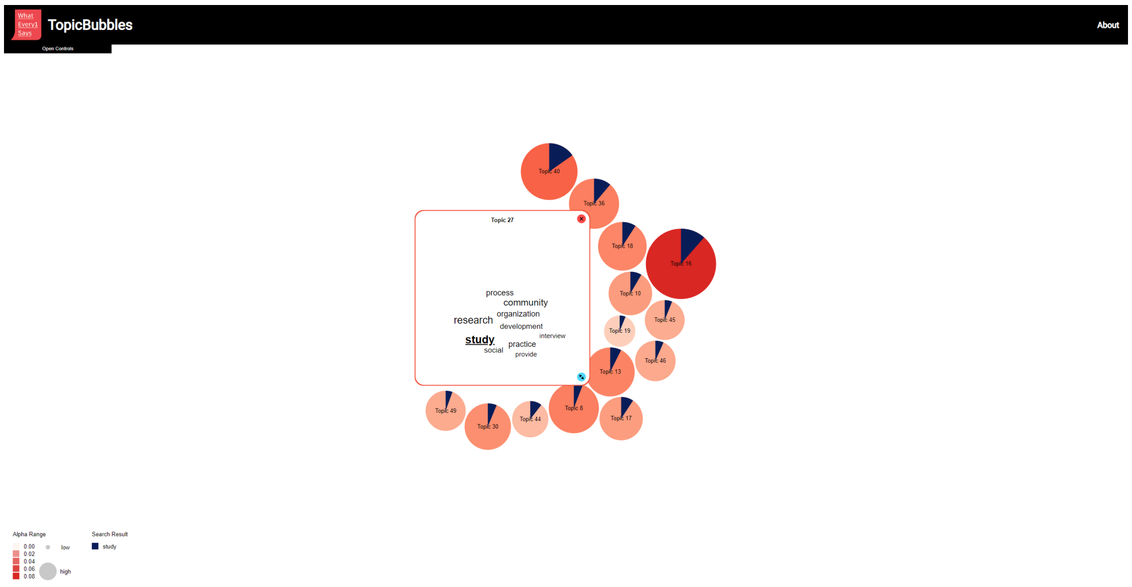


Figure 6.17: Clicking the word in the word cloud works as a filter to show only the topic bubble that contains the word

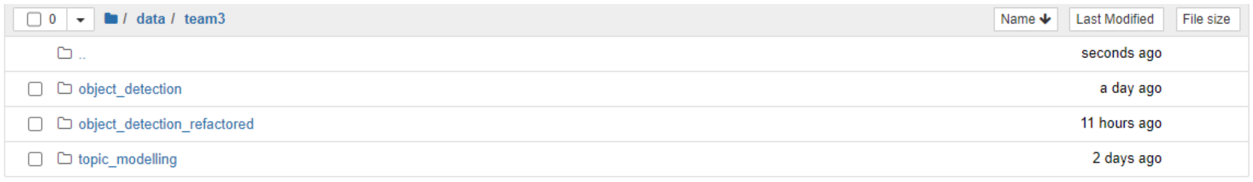
# Chapter 7

# Developer Manual

In this chapter, we describe the onboarding steps which any interested developer can go through in order to be able to contribute to this project.

## 7.1 Folder Structure

Navigating to the Team 3 folder in the cloud through "/tree/camelot/team3/" (see Figure 7.1) will show the user the top level topic modeling and object detection directories, which contain the code used to perform the respective tasks. For object detection, the team has two directories where object_detection is the folder with the APIs and Flask application, which contains the post-processing rules that we have come up with over the semester, as well as the code which utilizes Team 1's API to upload detected objects. The other folder, object_detection_refactored, contains the cleaned and refactored versions of both the Faster R-CNN and YOLOv7 models, which are to be used going forward. The code repository is organized such that it is easier to add support for other object detection models in the future. There are still filtering rules and false positives that have not been addressed with the refactored model. In the future it would help to add the post-processing rules that were implemented in the original model (explained in Section 4.3.1), as there are only the necessary rules to detect objects and build an XML structure, in the refactored version. Both folders have the capability to run inference with either Detectron2 or YOLOv7.



| Name | Last Modified |
|---|---|
| .. | seconds ago |
| object_detection | a day ago |
| object_detection_refactored | 11 hours ago |
| topic_modelling | 2 days ago |

Figure 7.1: Folder structure for Team 3

## 7.2 Object Detection

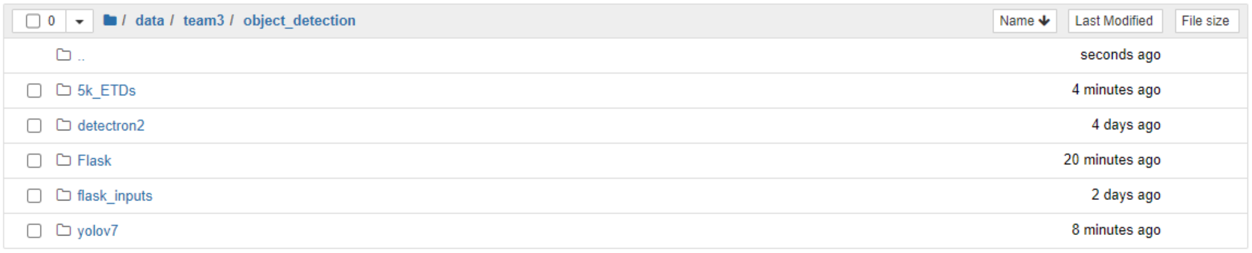
### 7.2.1 Access

A new user needs to have access to the following:

1. VT Computer Science Cloud [23]
2. Team 3 container cluster [25]
3. The GitLab repository for our Flask application with CI/CD

### 7.2.2 Steps

To access the container cluster, you must first get the access code from the VT CS Cloud website [23], illustrated in Figure 7.2 and explained below.



/ data / team3 / object_detection

| Name | Last Modified | File size |
|---|---|---|
| .. | seconds ago | |
| 5k_ETDs | 4 minutes ago | |
| detectron2 | 4 days ago | |
| Flask | 20 minutes ago | |
| flask_inputs | 2 days ago | |
| yolov7 | 8 minutes ago | |

Figure 7.2: Folder structure for object detection

1. 5k_ETDs: This folder contains the files used to perform the batch upload of objects to the Team 1 database.
2. detectron2: This folder contains the Detectron2 model.
3. Flask: This folder contains the Experimenter UI application.
4. flask_inputs: This folder contains the inputs that the Experimenter UI application uses.
5. yolov7: This folder contains the YOLOv7 model.

**Saving Objects**

1. Open the "/tree/camelot/team3/object_detection/5k_ETDs" directory in the container.

2. Run the code below in a notebook. The "etd_id" can be replaced by the PDFs from Team 1's database by running the curl command and GET request using Team 1's API.

```
!curl --output tmp/{etd_id}.pdf --request GET
https://team-1-flask.discovery.cs.vt.edu/v1/etds/{etd_id}/pdf

etd = f"tmp/{etd_id}.pdf"

!python ETDExtraction.py --source $etd --output "tmp"
```

3. After you run the cell, the objects are extracted and tagged, and they will be saved in the "tmp" folder mentioned in the code above.

**Running Flask**

1. Navigate to the "/tree/camelot/team3/object_detection/Flask/xml_parser/" directory in the container.

2. Run the following commands to run the Flask app:

```
$ export FLASK_APP=flask_main.py
$ flask run --host=0.0.0.0 -p 5001
```

**Services**

1. We have packaged our code as multiple services provided to Team 5, to be used for workflow automation, and for our front-end web page. A sample service (for our YOLOv7 model) is available at Docker Hub. Use the following URL to see how the service is to be used.

```
https://hub.docker.com/layers/nirmal2519/yolo/0.4/images/
sha256-64754d9407c042dc6f277e3261fc46fbeb8bc0f9fc78ecd79417f51927d173c3?
context=explore
```

2. Run the Docker image, and execute the following command:

```
$ python ETDExtraction.py \
        --source test/CMU-ISR-16-113.pdf \
        --output test
```

3. There is a sample ETD (`CMU-ISR-16-113.pdf`) in the Docker image for testing. Once this command is executed, the outputs as shown in Table 4.1 would be generated in the root directory for the sample ETD.

## 7.3 Topic Modeling

For Topic Modelling, we will be using the ETD Dataset to train various models in a Jupyter notebook which will generate pickle files corresponding to each model. Now, these model files will be referred to by the Flask application to serve APIs or display the results to a user's queries. For a developer to start contributing, one needs to follow the steps mentioned in the following sections.

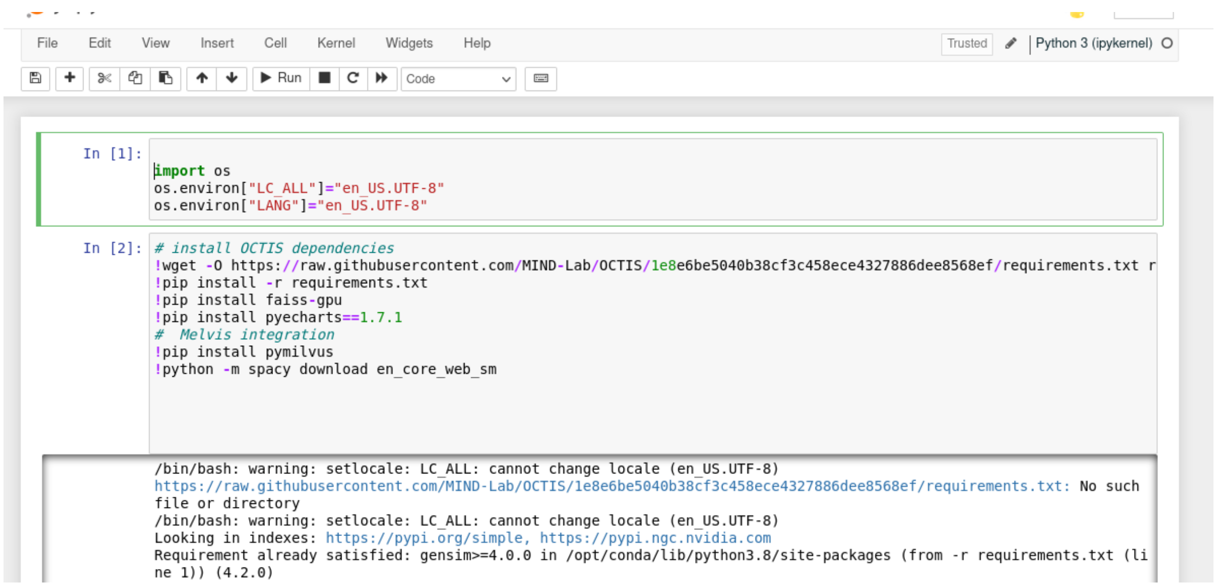
### 7.3.1 Access

A new user needs to have access to the following tools:

1. VT Computer Science Cloud [23]
2. Team 3 Topic Modeling container cluster [26]
3. The Git repository for our Flask application

### 7.3.2 Steps

1. Look for the ETD dataset stored in the "/tree/camelot/team3/topicmodeling/5k_ETDs" directory in the container. This file will be available in a CSV format. In this work we used the dataset containing 5000 entries which was available both locally at the above path and through API call from Team 1. Note that in the future, a bigger dataset can be used as appropriate.



```
In [1]: import os
        os.environ["LC_ALL"]="en_US.UTF-8"
        os.environ["LANG"]="en_US.UTF-8"

In [2]: # install OCTIS dependencies
        !wget -O https://raw.githubusercontent.com/MIND-Lab/OCTIS/1e8e6be5040b38cf3c458ece4327886dee8568ef/requirements.txt r
        !pip install -r requirements.txt
        !pip install faiss-gpu
        !pip install pyecharts==1.7.1
        #  Melvis integration
        !pip install pymilvus
        !python -m spacy download en_core_web_sm
```

```
/bin/bash: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)
https://raw.githubusercontent.com/MIND-Lab/OCTIS/1e8e6be5040b38cf3c458ece4327886dee8568ef/requirements.txt: No such
file or directory
/bin/bash: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)
Looking in indexes: https://pypi.org/simple, https://pypi.ngc.nvidia.com
Requirement already satisfied: gensim>=4.0.0 in /opt/conda/lib/python3.8/site-packages (from -r requirements.txt (li
ne 1)) (4.2.0)
```

Figure 7.3: A snapshot of topic_model.ipynb

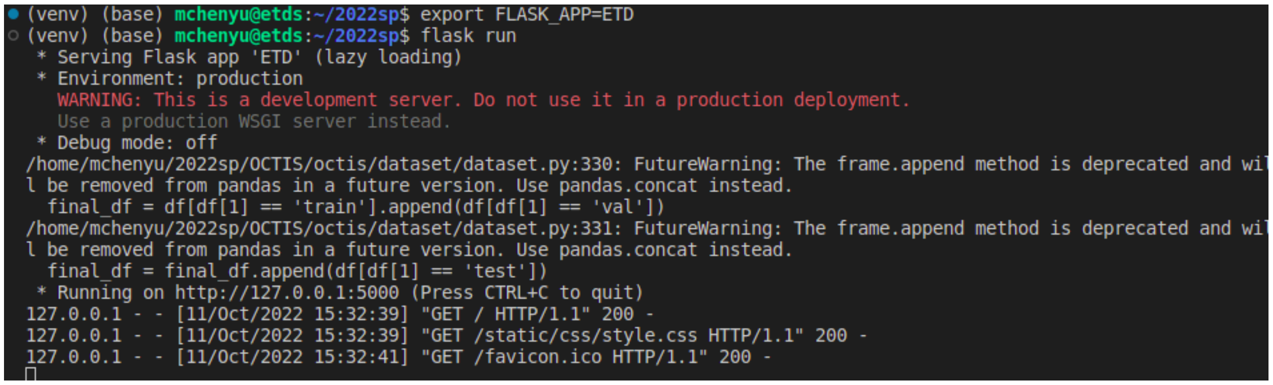
2. Open the topic_model.ipynb Python notebook location at path "data/team3/topic_modelling/TextMining/Code" in the container as shown in Figure 7.3.

3. By running all the cells, all the pre-requisites will be downloaded. Then the pre-processing steps and clustering will be done to generate the required output.

4. To make changes, git checkout to a new branch and add your code to the notebook.

5. After validating your code, add the new changes to ETD.py, which is the Flask application running the backend logic for the webview. Once the model is generated, it's time to see whether the Flask application is giving the correct webpages. To verify, use the command **flask run** to check on your local host. If no errors are found, the command line output will look like what appears in Figure 7.4.

6. Once the complete pipeline (from the model training code through the Flask application code) is working as intended, raise a pull request on the Git with detailed commit messages.



```
● (venv) (base) mchenyu@etds:~/2022sp$ export FLASK_APP=ETD
○ (venv) (base) mchenyu@etds:~/2022sp$ flask run
 * Serving Flask app 'ETD' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
/home/mchenyu/2022sp/OCTIS/octis/dataset/dataset.py:330: FutureWarning: The frame.append method is deprecated and wil
l be removed from pandas in a future version. Use pandas.concat instead.
  final_df = df[df[1] == 'train'].append(df[df[1] == 'val'])
/home/mchenyu/2022sp/OCTIS/octis/dataset/dataset.py:331: FutureWarning: The frame.append method is deprecated and wil
l be removed from pandas in a future version. Use pandas.concat instead.
  final_df = final_df.append(df[df[1] == 'test'])
 * Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
127.0.0.1 - - [11/Oct/2022 15:32:39] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Oct/2022 15:32:39] "GET /static/css/style.css HTTP/1.1" 200 -
127.0.0.1 - - [11/Oct/2022 15:32:41] "GET /favicon.ico HTTP/1.1" 200 -
```

Figure 7.4: Command line output of Flask application

# Bibliography

[1] Virginia Tech. "VTechWorks." *ETDs: Virginia Tech Electronic Theses and Dissertations*. 2011. Last Accessed December 12, 2022. https://vtechworks.lib.vt.edu/handle/10919/5534

[2] Uddin, Sami, et al. "Building a Large Collection of Multi-Domain Electronic Theses and Dissertations." *2021 IEEE International Conference on Big Data (Big Data)*. 2021.

[3] Alexandrova, Sonya, et al. "RoboFlow: A flow-based visual programming language for mobile manipulation tasks." *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015.

[4] Ahuja, Aman, et al. "Parsing Electronic Theses and Dissertations Using Object Detection." *The First Workshop on Information Extraction from Scientific Publications, WIESP 2022*. Taipei and Online, November 20, 2022, ACL.

[5] Ren, Shaoqing, et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, 2017, pp. 1137–1149*. 2017.

[6] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014.

[7] Girshick, Ross. "Fast R-CNN." *Proceedings of the IEEE International Conference on Computer Vision*. 2015.

[8] Wu, Yuxin, et al. "Detectron2." 2019. Last Accessed December 12, 2022. https://github.com/facebookresearch/detectron2

[9] Wang, Chien-Yao, et al. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors." *arXiv preprint arXiv:2207.02696*. 2022.

[10] Blei, David M., et al. "Latent Dirichlet Allocation." *Journal of Machine Learning Research. 3 Jan (2003): 993-1022*. 2003.

[11] Srivastava, Akash, et al. "Autoencoding Variational Inference for Topic Models." *arXiv preprint arXiv:1703.01488* (2017).

[12] Bianchi, Federico, et al. "Pre-Training Is a Hot Topic: Contextualized Document Embeddings Improve Topic Coherence." *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing. (Volume 2: Short Papers)*. 2021.

[13] Paszke, Adam, et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 2019.

[14] Lin, Tsung-Yi, et al. "Microsoft COCO: Common Objects in Context." *European Conference on Computer Vision*. 2014.

[15] Belval. "Belval/pdf2image: A Python Module That Wraps the pdftoppm Utility to Convert PDF to PIL Image Object." *GitHub*. Last Accessed December 12, 2022. Last Accessed December 12, 2022. https://github.com/Belval/pdf2image

[16] Grinberg, Miguel. "Flask Web Development: Developing Web Applications with Python." *O'Reilly Media, Inc.*, 2018.

[17] Eby, P. J. "Python Enhancement Proposals." *PEP 3333 – Python Web Server Gateway Interface v1.0.1*. 2010. Last Accessed December 12, 2022. https://peps.python.org/pep-3333/

[18] Terragni, Silvia, et al. "OCTIS: Comparing and Optimizing Topic models is Simple!" *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. 2021.

[19] Jia, Yangqing, et al. "Caffe: Convolutional Architecture for Fast Feature Embedding." *Proceedings of the 22nd ACM International Conference on Multimedia*. 2014.

[20] Docker, Inc. "Docker." 2013. Last Accessed December 12, 2022. https://www.docker.com/

[21] Python Software Foundation. "Python." 2001. Last Accessed December 12, 2022. https://www.python.org/

[22] Collobert, Ronan. "Torch." 2017. Last Accessed December 12, 2022. https://github.com/torch/torch7

[23] Virginia Tech. "Computer Science Cloud." 2022. Last Accessed December 12, 2022. https://cloud.cs.vt.edu/

[24] Virginia Tech. "VT GitLab Repository." 2022. Last Accessed December 12, 2022. https://code.vt.edu/

[25] Virginia Tech. “Object Detection Container.” 2022. Last Accessed December 12, 2022. https://team3-container-1.discovery.cs.vt.edu/

[26] Virginia Tech. “Topic Modeling Container.” 2022. Last Accessed December 12, 2022. https://team3-container-2.discovery.cs.vt.edu/

[27] GitLab. “GitLab CI/CD.” 2011. Last Accessed December 12, 2022. https://docs.gitlab.com/ee/ci/