Object Detection and Topic Modeling

Dr. Edward Fox
Virginia Tech, Blacksburg VA 24061
Dec. 1, 2022

CS5604 Team 3

Nila Masrourisaadat, Raj Sahu, Alan Devera, Nirmal Amirthalingam, Chenyu Mao
Overview

● 5 teams were tasked to build an information storage and retrieval system from scratch to make Electronic Theses and Dissertations (ETDs) more accessible to the researchers, experimenters, and curators.

● Our team (Team 3) was tasked with detecting objects (Object Detection) within ETDs as well as determining topics (Topic Modeling) to store in a repository (Team 1) for better search and recommendation of ETDs (Team 2).
Overall Pipeline

ETD data

Tokenization
(Title and abstract of each PDF)

Clustering using OCTIS

CTM
LDA

Neural LDA
ProdLDA

List of k-topics with associated docs

ETD Embeddings
Related Documents
Related Topics

Object Detection

Faster R-CNN
YOLO

Extracted ETD objects

Post-processing

Segmented Chapters for Topic Browser Integration

Image and Text Objects

Experimenter webpage
Data Structure for ETD objects
Data Structure for ETD hierarchy

Repository (Team 1)

XML
Object Detection

- Object Detection overview
- ETD filtering rules
- Issues and challenges faced
- Services
- 5k ETDs
- Demo
- Future work
Object Detection Pipeline
Tools Used

- Pdf2image
- PyMuPdf
- YOLOv7
- Detectron2 (Faster R-CNN)
- Flask
- Docker
- VT CS Cloud
YOLOv7 & Detectron (Faster R-CNN)

**YOLOv7:**
- A real-time object detector that has greatly advanced the CV & ML world
- Fastest, and most accurate object detector to date
- Uses CNN to predict bounding boxes and class probabilities considering the entire image at one step

**Detectron2 :**
- A framework for object detection built on top of PyTorch
- Has support for object detection, activity recognition, semantic/instance segmentation
- Trains an object detection model on custom datasets using pre-trained weights
- All models in Detectron2 are pre-trained on the COCO dataset

***For our experiments, we would be using faster R-CNN models present in Detectron2***
XML Schema

- Each PDF has its ETD ID as the root element
- 3 sub-elements - front, body, and back
- Image-based objects
  - Figures
  - Tables
  - Equations
  - Algorithms
- Image-based objects have captions / numbers associated with them
- The rest are text-based objects
False positives

Problem:

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

Solution:

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

Problem:

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

Solution:

- Determined the relationship between figure/table and their corresponding caption
- Paired images with their corresponding captions according to them being on top or bottom of images

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.

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.
False positives

Problem:

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

Solution:

- Checked if two consequent chapter titles are the same
ETD Filtering Rules

Post-processing rules are as follows:

1. Linked image object with the previous caption object in the list
   a. Determined the relationship between figure/table and their corresponding caption
   b. Paired images with their corresponding captions according to them being on top or bottom of images
   c. Checked the orientation of the pages in case of being horizontal and changed to vertical

2. Filtered chapters/sections titles
   a. Checked the first letter of the title (capital)
   b. Checked if the y-coordinate of the bounding box is within half of the page’s height
   c. Avoided the text overflow for consecutive pages by checking the punctuation
   d. Checked if two consequent chapter titles are the same
   e. Created null section tag in case of titles not being detected

3. Eliminated false positives based on matching with ToC
   a. Extracted and saved all the detected chapter/section titles, ToC, page numbers
   b. Filtered out the chapter and section titles that have been incorrectly detected

4. Refine chapter title detection by removing outliers
   a. Find if the keyword “chapter” is in the list of all detected chapter titles
   b. Difference in font size between chapter titles and other objects
   c. Check the indentation level - left / center alignment
Team 3 Deliverables - Objects

Discussing with the other teams, and assessing the overall requirements of the class, we are providing the following outputs:

<table>
<thead>
<tr>
<th>Output for a given ETD</th>
<th>Description/Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page images</td>
<td>Each page of ETD is saved as a .jpg file</td>
</tr>
<tr>
<td>Detected images</td>
<td>Consists of sub-folders for each image-based object class (figure, table, equation, algorithm) saved as .jpg files</td>
</tr>
<tr>
<td>JSON object</td>
<td>Each object - text and image-based, is saved (unordered set as per the detection sequence)</td>
</tr>
<tr>
<td>Parsed XML</td>
<td>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</td>
</tr>
</tbody>
</table>
We are providing 3 services that are containerized - to be used for workflow automation and our frontend webpage

- Generate page images given an ETD ID
- Generate outputs given an ETD ID - using YOLOv7
  - Detected images
  - JSON objects
  - XML
- Generate outputs given an ETD ID - using Faster R-CNN (Detectron2)
  - Detected images
  - JSON objects
  - XML
5k ETDs - Inference and Storing in DB

- Currently using the save detected object API (from team 1) to store objects in DB
- Finished running on 5000 ETDs
- Average inference time ~ 3 minutes (GPU)

Save detected object (POST) API -

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

- <etd_id>/page
- <etd_id>/text
- <etd_id>/image
- <etd_id>/xml
# 5k ETDs - Inference and Storing in DB

<table>
<thead>
<tr>
<th>Type</th>
<th>File</th>
<th>Metadata</th>
</tr>
</thead>
<tbody>
<tr>
<td>page</td>
<td><code>&lt;page_number&gt;.jpg</code></td>
<td>-</td>
</tr>
<tr>
<td>text</td>
<td>-</td>
<td><code>{keys = type, text, page_num, bbox}</code></td>
</tr>
<tr>
<td>image</td>
<td><code>&lt;etd_id&gt;_&lt;class&gt;_&lt;count&gt;.jpg</code></td>
<td><code>{keys = type, path, page_num, bbox}</code></td>
</tr>
<tr>
<td>xml</td>
<td><code>&lt;etd_id&gt;.xml</code></td>
<td>-</td>
</tr>
</tbody>
</table>

Get objects by ETD ID API

https://team-1-flask.discovery.cs.vt.edu/v1/etds/<etd_id>/objects?type=<type_name>
Experiment UI Choosing a model

2 Models:

- YOLOv7
- Detectron2 (Faster R-CNN)
Experimenter UI Upload ETD

- Experimenter has the option to upload a pdf of the ETD that they want to perform object detection on.
Experimenter UI ETD Browser

- Sidebar to navigate to different chapters and sections
- Images are linked to their respective captions
See the video demonstration using the file ETDViewerDemo.mp4
Milestones/Timeline

- Work done till IR-2
  - Implemented basic XML schema for detected objects
  - Developed post-processing rules/filters for YOLOv7
  - Front-end wireframes for the Experimenter Web Page

- Work done for IR-3
  - Added support for Faster R-CNN
  - Finalized working models and parser logic for YOLOv7 and Faster R-CNN (base models)
  - Deployed both models on the cloud server (containerized)
  - Team 1 deliverables
  - Working prototype of the Experimenter Web Page using Flask that supports both models

- Work done in November and December
  - Experiment with a bigger subset of the ETD dataset
  - Run inference on 5k ETDs using team 1’s API to store into their DB
  - Dockerized services for object detection and enabled CI/CD features for the frontend
Future Work - Object Detection

- Integrate Flask app / Experimenter UI with the frontend and workflow services
- Update our pipeline to support “Add an ETD” once team 1 provides the API for saving metadata
- Set up workflow automation - team 5
- Add support for scanned documents - modify existing pipeline
- Add more post processing rules based on the results of the 5k dataset
- Improve UI layout and add more functionality
Topic Modeling

- Background
- Pipeline
- Chapters
- Experimenter pages + Demo
- Services
- Milestones
- Wrap-Up Tasks
Background - **Vector Distance**

- **Input ETD -> Octis -> Vectors**
- **Vectors** [0.31, 0.47, 0.02, …, 0.21]
- **Shape**: (1, 50)

- **Similarity** = Euclidean Distance of two vectors

TM - Pipeline

- Dataset preprocessing
- Topic Model training
- Inference Learning
- Topic and Documents Visualization
## TM - Deliverables

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Related Topics &amp; Related Docs</td>
<td>For a given ETD id or Chapter id, find the most similar topics and documents respectively.</td>
</tr>
<tr>
<td>Topic Browser (Experimenter)</td>
<td>Allow a user to browse all topics, click on one and read associated documents. (Documents by Topic - Chapters by Topic)</td>
</tr>
</tbody>
</table>
TM - Chapters - Initial Experiment

- Extract all <para> tags to form a chapter from an ETD’s XML.
Old Table schema:

<table>
<thead>
<tr>
<th>id</th>
<th>Title</th>
<th>Abstract</th>
<th>Author</th>
<th>Year</th>
<th>University</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ETD Title 1</td>
<td>ETD Abstract1</td>
<td>A1,A2</td>
<td>1900</td>
<td>Virginia Tech</td>
</tr>
<tr>
<td>2</td>
<td>ETD Title 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ETD Title 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
New Table schema:

<table>
<thead>
<tr>
<th>id</th>
<th>Type</th>
<th>Parent</th>
<th>Title</th>
<th>Abstract</th>
<th>Author</th>
<th>Year</th>
<th>University</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ETD</td>
<td>NULL</td>
<td>ETD Title 1</td>
<td>ETD Abstract 1</td>
<td>A1,A2</td>
<td>1900</td>
<td>Virginia Tech</td>
</tr>
<tr>
<td>2</td>
<td>ETD</td>
<td>NULL</td>
<td>ETD Title 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ETD</td>
<td>NULL</td>
<td>ETD Title 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Chapter</td>
<td>1</td>
<td>Chap Title 1</td>
<td>Chap Abstract 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Chapter</td>
<td>1</td>
<td>Chap Title 2</td>
<td>Chap Abstract 2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TM - Chapters - Initial Experiment - **Issues**

Dataset size → 10x (Assuming each ETD has 10 Chapters)

For same memory, reduce Overall number of ETDs and the vocabulary size.

Reduced quality of topic categorization and search
TM - Chapters - **Inference**

![Diagram showing a trained model with topic-document-matrix and topic-word-matrix inputs, connected to segmented chapters.](image-url)
TM - Chapters - Inference - **Related Topics**
# convert word list into N-hot encoding where total array size is length of vocabulary
terms_idx=[word2idx.get(w, None) for w in words_list]
query_vector=np.zeros(len(word2idx))
count = 0
for idx in terms_idx:
    if idx is not None:
        query_vector[idx]=1.0
count+=1

## Related Topics
# Using cosine similarity, find the most similar looking vocabulary array
# compared to our N-hot encoded words list
query_vector = np.expand_dims(query_vector, axis=0)  # convert to 2d for cosine similarity
distances = cosine_similarity(query_vector, model['topic-word-matrix'])
matching_topics_idx = np.argsort(-distances)
matching_topics_idx = np.squeeze(matching_topics_idx)  # decreasing order of most similar topic ids
logger("Related Topics : ",matching_topics_idx[:topk],2)
# convert word list into N-hot encoding where total array size is length of vocabulary
terms_idx=[word2idx.get(w, None) for w in words_list]
query_vector=np.zeros(len(word2idx))
count = 0
for idx in terms_idx:
    if idx is not None:
        query_vector[idx] = 1.0
        count += 1

## Related Topics
# Using cosine similarity, find the most similar looking vocabulary array
# compared to our N-hot encoded words list
query_vector = np.expand_dims(query_vector, axis=0)  # convert to 2d for cosine similarity
distances = cosine_similarity(query_vector, model['topic-word-matrix'])
matching_topics_idx = np.argsort(-distances)
matching_topics_idx = np.squeeze(matching_topics_idx)  # decreasing order of most similar topic ids
logger("Related Topics : ", matching_topics_idx[:topk], 2)
# convert word list into N-hot encoding where total array size is length of vocabulary
terms_idx=[word2idx.get(w,None) for w in words list ]
query_vector=np.zeros(len(word2idx))
count = 0
for idx in terms_idx:
    if idx is not None:
        query_vector[idx]=1.0
        count+=1

### Related Topics
# Using cosine similarity, find
# compared to our N-hot encoded
query_vector =np.expand_dims(query_vector,d =0) # convert to 2d for cosine similarity
distances=cosine_similarity(query_vector,mat=p['topic-word-matrix'])
matching_topics_idx = np.argsort(-distances)
matching_topics_idx = np.squeeze(matching_topics_idx) # decreasing order of most similar topic ids
logger("Related Topics : ",matching_topics_idx[:topk],2)
Calculating similarity

# convert word list into N-hot encoding where total array size is length of vocabulary
terms_idx=[word2idx.get(w,None) for w in words_list ]
query_vector=np.zeros(len(word2idx))
count = 0
for idx in terms_idx:
    if idx is not None:
        query_vector[idx]=1.0
    count+=1

# Using cosine similarity, find the most similar looking vocabulary array
# compared to our N-hot encoded words list
query_vector =np.expand_dims(query_vector, axis=0) # convert to 2d for cosine similarity
distances=cosine_similarity(query_vector,model['topic-word-matrix'])
matching_topics_idx = np.argsort(-distances)
matching_topics_idx = np.squeeze(matching_topics_idx) # decreasing order of most similar topic ids

logger("Related Topics : ",matching_topics_idx[1:10],2)
TM - Chapters - Inference - **Related ETDs**

- Segmented Chapters → Cleaning → Query

  - **Trained Model**
    - topic-document-matrix
    - topic-word-matrix

  - Cosine Similarity → Top-5 Topics

  - For each topic id → Top-5 ETDs
### Related etds

```python
# iterate on all related topics, and fetch top-k related ETDs based on each. Sort and find out overall top-5

topic_document_matrix = model['topic-document-matrix']
list_of_competitors = []

for topic_id in matching_topics_idx[:topk]:
    local_toppers_id = (-topic_document_matrix[topic_id]).argsort()[:topk].tolist()
    for etd_id in local_toppers_id:
        probability_value = topic_document_matrix[topic_id][etd_id]
        list_of_competitors.append([-probability_value, etd_id])  # appending -ve so that sorting gives the desired
list_of_competitors.sort()
matching_etd_idx = [etd_id for _,etd_id in list_of_competitors[:topk]]

logger("Related ETDs : ", matching_etd_idx,2)
```

Top-5 topic ids
### Related etds

# iterate on all related topics, and fetch top-5 etds related to each. Sort and find out overall top-5

topic_document_matrix = model['topic-document-matrix']

list_of_competitors = []

for topic_id in matching_topics_id[:topk):
    local_toppers_id = (-topic_document_matrix[topic_id]).argsort()[:topk].tolist()
    for etd_id in local_toppers_id:
        probability_value = topic_document_matrix[topic_id][etd_id]
        list_of_competitors.append([probability_value, etd_id])  # appending -ve so that sorting gives the desire

list_of_competitors.sort()

matching_etd_idx = [etd_id for _, etd_id in list_of_competitors[:topk]]

logger("Related ETDs : ", matching_etd_idx, 2)
TM - Chapters - Inference - Issues

- Quality of output $\propto$ Accuracy of Segmentation
Experimenter pages - **Topic Bubbles**

- Intuitive and Interactive visualization

Source: https://github.com/sihwapark/topic-bubbles
tw.json

topic_weights {
  alpha : [ a_1, a_2, ..., a_t ],
  tw : [
    {words : [ w_1, w_2, ..., w_n ],
      weights : [ f_1, f_2, ..., f_n ] }
  ]
}
## meta.csv

<table>
<thead>
<tr>
<th>author</th>
<th>title</th>
</tr>
</thead>
<tbody>
<tr>
<td>name₁</td>
<td>&quot;Titus, Monica Joy&quot;, &quot;Plasma Diagnostics and Plasma-Surface Interactions in Inductively Coupled Plasmas&quot;</td>
</tr>
<tr>
<td>name₂</td>
<td>&quot;Condie, Tyson&quot;, &quot;Declarative Systems&quot;</td>
</tr>
<tr>
<td>name₃</td>
<td>&quot;Horberg, Elizabeth Jana&quot;, &quot;Portrait of the Rugged Individualist: The Nonverbal Pride Display Communicates Support for Meritocracy&quot;</td>
</tr>
<tr>
<td>etd₁</td>
<td>&quot;Nelson Mondragon, John Alexander&quot;, &quot;Essays in Empirical Macroeconomics&quot;</td>
</tr>
<tr>
<td>etd₂</td>
<td>&quot;Swift, Timothy Alan&quot;, &quot;Control and Trajectory Generation of a Wearable Mobility Exoskeleton for Spinal Cord Injury Patients&quot;</td>
</tr>
<tr>
<td>etd₃</td>
<td>&quot;Leveille Buchanan, Nicole Therese&quot;, &quot;Errors as a Productive Context for Classroom Discussions: A Longitudinal Analysis of Nair, Pradeep&quot;, &quot;Interrogating the Role of Spatial Organization in Receptor Function: Eph-Ephrin Signaling in Breast Cancer&quot;</td>
</tr>
</tbody>
</table>

- "Sardjono, Sandra", "Tracing Patterns of Textiles in Ancient Java (8th-15th century)"
- "Payal, Anuj Ashwin", "Microtopographical control of cell adhesion, organization, and proliferation in a cardiac tissue engineered tissue"
- "Elias, Renee Roy", "Grocery Stores: Neighborhood Retail or Urban Panacea? Exploring the Intersections of Federal Policy, Commerce"
dt.json

topic_documents {
  topic_1 : [ id_1, id_2, ..., id_20 ],
  ...
  topic_50 : [ id_1, id_2, ..., id_20 ],
}
TM - Services

We are providing 2 services that will be containerized - to be used for workflow automation and our frontend webpage

- Given a dataset, train TM model to generate vectors.
- Generate related topics and documents for an ETD and all its chapters.

Ps: The Frontend flask application (hosted on Cloud CS) is not linked with service yet.
Milestones/Timeline

● Work done till IR-2
  ○ Train on complete 500k ETD dataset and integrate with UI
  ○ Perform similarity search (topic+document) through FaisNN
  ○ Display top-5 topics when topic search query is used

● Work done for IR-3
  ○ Chapter dataset generation, pre-processing
  ○ Integrating chapters into topic model
  ○ Preprocessing chapter data for OCTIS
  ○ Code refactoring and optimization for reducing load time

● Work done for Final Report
  ○ Topic Bubble Experimenter UI
  ○ Generate and save Related topics+Docs data for 5k dataset for all ETDs and Chapters
Wrap-Up Tasks

- Set up workflow automation - team 5
- Integrate Flask app / Experimenter UI with the frontend and workflow services.
- Build the Bubble UI for all the models (CTM, LDA, NeuralLDA and ProdLDA)
QUESTIONS?
BACKUP
Object Detection

- What has been accomplished
- Deliverables
- ETD filtering rules
- Issues and challenges faced
- Milestones
- Upcoming tasks and goals
What Has Been Accomplished

● The basic XML schema has been implemented for all object types
● Retrieved a trained YOLOv7 model to run inferences
● Post-processing rules for YOLOv7 detections
● Experimented on 10 randomly sampled PDFs for testing
● Front-end wireframe (Experimenter and page after search)
● Docker container with GPU access enabled
What Has Been Accomplished

- Set up Docker containers for YOLOv7 and Detectron2 frameworks
- Implemented YOLOv7 and Faster R-CNN algorithms for ETDs
- Implemented post-processing rules for detected objects
- Converted the unordered set of detections to XML
- Experimented on multiple randomly sampled ETDs
- Experimenter Web Page UI
- Team 1 deliverables - DB tasks (read/write)
Issues / Test Cases - OD

- Detections are not always in the top-bottom order for pages which is required to create the XML tree
Front sub-element

Title: Probabilistic Models of Topics and Social Events

Author: CMU-IRIM-16-113

University: Carnegie Mellon University

Committee: Thesis Committee: Unpublished

Abstract: Structured probabilistic inference has shown to be useful in modeling complex element structures of data. One successful way in which this technique has been applied 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 social networking, we can now easily acquire text data attached to meta-information, such as geo-spatial coordinates and time stamps. This meta-data can provide rich information that is helpful in answering many research questions related to spatial and temporal reasoning. However, such data must be treated differently than text data. For example, spatial data is usually organized in terms of a two-dimensional region while temporal information can exhibit periodicities. While some work existing in the topic modeling community that utilizes some of these meta-information, 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 towards two directions: First, I augment the model using recurrent Chinese restaurant process (RCPP) to discover events that are dynamic in nature. Second, I studied a model that can detect events using data from multiple media sources. I studied the characteristics of different media in terms of spread of events and linguistic patterns. The approaches studied in this thesis are largely based on Bayesian non-parametric methods to deal with streaming data and unpredictable number of clusters. The research will not only serve the event detection problem itself but also bring light into a more general structure clustering problem in spatial, temporal and textual data.
A naive approach to initialize the latent variables is to use uniformly generated random variables to serve as the initial values of $z$ and $s$. However, if those initial values are bad, it is likely that the algorithm would take a long time to reach equilibrium. Instead, we can sample the initial $z$-q values and using parts of Equation 5.8 and Equation 5.4 that do not require our knowledge to cluster indices $s$. For location index $i$, its initial value is sampled purely based on its location proximity to the Gaussian centers. For word category variable $v$, its initial values are determined by both the values of previous words in the current document and prior $\pi$.
Front End - Object Detection

- End product - Experimenter web page
Abstract

Do so anaphors are a fairly widely used in English, but has received relatively little treatment in the literature (especially when compared with verb phrase ellipses). 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 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 anaphors has been used to argue both hierarchical structure (Lukoff and Ross 1976) and flat structure within the verb phrase (Curfman and Jacobstorf 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 Manzanares and Tag (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. A 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 present a novel analysis that utilize three semantic parameters (agentivity, addressee, and definiteness) 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 their contain do so in a non-finite form (usually in the infinitive), and in others, the antecedent 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 ellipses 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 of 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 Introduction

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 conjunct 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 occurred as that expressed in the first clause. These verbal anaphors include Verb Phrase Ellipsis (2a), do so 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.

---

End product - Parsed ETD web page
Upcoming Tasks and Goals - OD

● Link image-based objects to their captions based on a distance metric (like Euclidean)
  ○ Figures - figure captions
  ○ Tables - table captions
  ○ Equations - equation numbers
● Recognizing the right chapter and section titles (delimiters)
● Fix errors and debug
● Deal with subsections and sections as they constitute the same object / class
● Building the web pages (Experiment + Parsed ETD)
Issues/Test Cases - OD

False positives:

1. Chapter/section titles being incorrectly detected
   a. Paragraph’s last line
   b. New chapter tag for the same chapter title
   c. Chapter and paragraph is being created but not the section

2. Images/tables being linked to wrong caption

3. Chapter/section titles detected don’t match with the titles in the ToC
3.3.1 Event Model

An important observation incorporated into our model is that events are in many ways natural extensions of topics; events have a topical focus but also include a spatial and temporal region in which they are likely to occur. We thus assume events are defined by three things: (1) each event has a geographical center as well as a geographical variance controlled by a diagonal σ(τ) covariance matrix with each value defined by the location of a document that belongs to a event is assumed to be drawn from a two-dimensional Gaussian distribution governed by these.
Experimenter UI - Faster R-CNN Detection

ETD Browser

Chapter 1 Introduction
Chapter 2 Background
Chapter 3 Modeling Independent Events
Chapter 4 Modeling Temporal Evolutive Events
Chapter 5 Modeling Events from Multiple Data Sources
Chapter 6 Conclusions and Future Work

Probabilistic Models of Topics and Social Events

University:
Degree:
Committee:
Date:

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 applied 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-called 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-dimensional region while temporal information can exhibit periodicities. While some work existing in the topic modeling community that utilizes some of the meta information, 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 nature. Second, I studied a model that can detect events using data from multiple media sources. I studied the characteristics of different media in terms of reported event times and linguistic patterns. The approaches studied in this thesis are largely based on Bayesian non-parametric methods to deal with streaming data and unpredictable number of classes. The research will not only serve the event detection problem itself but also shed light into a more general structured clustering problem in spatial, temporal and textual data.

Structured probabilistic inference has shown to be useful in modeling complex latent structures of data. One successful way in which this technique has been applied is in the discovery of latent topical structures of text data,
UI improvements

- Filter Layout
  - Year range
  - Author names
  - University
  - Faculty
  - Committee