Toward a Big Data Analysis System for Historical Newspaper Collections Research

Sandeep Puthanveetil Satheesan
University of Illinois at Urbana-Champaign
Urbana, Illinois, USA
sandeeps@illinois.edu

Bhavya
University of Illinois at Urbana-Champaign
Urbana, Illinois, USA
bhavya2@illinois.edu

Adam Davies
University of Illinois at Urbana-Champaign
Urbana, Illinois, USA
adavies4@illinois.edu

Alan B. Craig
Extreme Science and Engineering Discovery Environment
USA
alanbcraig@gmail.com

Yu Zhang
California State University, Fresno
Fresno, California, USA
yuzh@csufresno.edu

ChengXiang Zhai
University of Illinois at Urbana-Champaign
Urbana, Illinois, USA
czhai@illinois.edu

ABSTRACT

The availability and generation of digitized newspaper collections have provided researchers in several domains with a powerful tool to advance their research. More specifically, digitized historical newspapers give us a magnifying glass into the past. In this paper, we propose a scalable and customizable big data analysis system that enables researchers to study complex questions about our society as depicted in news media for the past few centuries by applying cutting-edge text analysis tools to large historical newspaper collections. We discuss our experience with building a preliminary version of such a system, including how we have addressed the following challenges: processing millions of digitized newspaper pages from various publications worldwide, which amount to hundreds of terabytes of data; applying article segmentation and Optical Character Recognition (OCR) to historical newspapers, which vary between and within publications over time; retrieving relevant information to answer research questions from such data collections by applying human-in-the-loop machine learning; and enabling users to analyze topic evolution and semantic dynamics with multiple compatible analysis operators. We also present some preliminary results of using the proposed system to study the social construction of juvenile delinquency in the United States and discuss important remaining challenges to be tackled in the future.

CCS CONCEPTS

• **Applied computing** → **Optical character recognition**; Document analysis; • **Computing methodologies** → *Image segmentation*; **Natural language processing**; **Information extraction**; *Neural networks*; • **Human-centered computing** → *Information visualization.*

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

KEYWORDS

Historical newspapers, Big data analysis system, Information retrieval, Text analysis, Natural language processing, Image analysis, Newspaper article segmentation, Data visualization, Social science research, Juvenile delinquency, Social construction

ACM Reference Format:
Sandeep Puthanveetil Satheesan, Bhavya, Adam Davies, Alan B. Craig, Yu Zhang, and ChengXiang Zhai. 2022. Toward a Big Data Analysis System for Historical Newspaper Collections Research. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Digitized historical newspaper collections create unique opportunities and challenges in the realm of big data analysis systems. For domains like the social sciences and journalism, the availability of these large data collections can provide researchers (and the public) with the resources to ask broader questions of historical relevance. This type of information, if available, was previously only accessible by visiting a library or an archive and performing extensive manual research.

Many new insights can be obtained by querying these large collections, which often span multiple centuries. The availability of large collections of digitized data is a significant step in retrieving answers to these questions, but it is only one such step. Recent years have seen significant progress in Natural Language Processing (NLP) and text analytics, creating an unprecedented opportunity to leverage advanced text analysis techniques to provide interactive support for social scientists to analyze massive amounts of text content to explore and test interesting research hypotheses. However, several complex challenges remain in developing a fully customizable and scalable big data analysis systems focused on processing such historical newspaper collections.

First, many of the available historical newspaper collections contain millions of digitized newspaper pages from various publications worldwide, which amount to hundreds of terabytes of data. This size of the data and the required real-time interaction create challenges in designing efficient, intelligent algorithms to quickly process the data and generate insightful semantic analysis results.

Even retrieving relevant information to answer research questions from such data collections, which are huge both in size and variety, presents challenges that are in many ways unique to newspapers in general and sometimes specific to historical newspapers.

Second, newspapers present many variations between publications and within publications over time. Though the text generated by performing OCR on historical newspapers has improved considerably over the years, it is far from perfect. Also, digitized newspaper collections do not usually segment articles or differentiate them from advertisements. Thus, much work must be done before optimally relevant materials may be retrieved.

Third, it is unclear how to design a unified system for flexible analysis. Although many NLP and image analysis tools are available, there are questions about which ones to employ, and how to integrate them. Additionally, customizing them to effectively support a researcher in exploring, generating, and testing interesting research hypotheses remains an open challenge. In particular, as it is often difficult to pre-specify the analysis process, we want to enable users to interactively control the analysis, combining multiple tools as needed. Moreover, different digitized historical newspaper collections and the data available through them may need different kinds of analysis, depending on the research question. So, how can we add novel capabilities to support dynamic analysis involving users in the loop?

In this paper, we study how to build such a system by leveraging state-of-the-art Artificial Intelligence (AI) techniques to address the challenges outlined above. Specifically, our contributions are as follows:

(1) We propose four design principles and a modularized architecture for creating such a system.
(2) We evaluate how well some of the current AI techniques (e.g., OCR, newspaper article segmentation, and semantic search) work on the historical newspaper images and text.
(3) We propose using multiple compatible text analysis operators to support a potentially huge number of different analysis workflows (to accommodate different application needs).
(4) We provide initial support for human-in-the-loop collaboration between social science researchers and AI-driven analysis systems.

We design a novel customizable and scalable big data analysis system for processing and analyzing digitized historical newspaper collections, implement a preliminary version of this system, and evaluate its utility using a case study of the social construction of juvenile delinquency in the U.S. Finally, we discuss our research findings, and elaborate the challenges that remain before it will be possible to fully realize our vision.

2 RELATED WORK

Historical big data analysis systems that use text and image data are an active area of research and development. Our system is related to some existing application systems in this area. Some of the initial works on creating searchable historical newspaper collections mainly involved the tireless manual effort of collecting, curating, and transcribing documents [9]. The Library of Congress (LOC) has recently designed and developed a web application called Newspaper Navigator, which is a public dataset that contains extracted headlines and visual content processed on 16.3 million historical newspaper pages on the Chronicling America (LOC-CA) website [25]. This is an extensive work involving multiple products, including a scalable pipeline, a fine-tuned Deep Learning model, and a website for querying and retrieving image and textual content. In many ways, this is close to the retrieval component in the proposed system. However, we need to go beyond a standard search capability by supporting interactive text analysis as many hypotheses require deeper analysis of text content (e.g., temporal pattern analysis, topic evolution analysis, and comparative semantic analysis). Such capabilities are generally not available in current systems dealing with newspaper content. Our system aims to support researchers in a general way, including a visualization interface to enable a user to to digest patterns in data intuitively.

Word embedding techniques have been recently shown to be quite effective for computing lexical relations and representing text data in a neural network [36, 47]. We use these techniques to provide interactive text analysis functions for social science researchers, enabling them to explore interesting lexical relations in historical newspaper collections. Such functions have been studied in the data mining community [49], where a topic cube is constructed to enable Online Analytical Processing (OLAP) of both text and non-text data (e.g., time and location). However, this kind of analysis has not yet been practical enough to be implemented in a robust interactive system, and usually only capture general topics and trends, which are insufficient for deeply analyzing a focused topic such as juvenile delinquency. More specialized text analysis systems have also been developed (e.g., for interactively analyzing reviews [50]), but no such system has been built that can analyze large historical newspaper collections.

There are already some text analysis software that can provide partial support for specific types of analyses, which we reuse as modules in our system. The ShiCo tool [23, 30] supports visualizing concept changes over time that we apply to support users in discovering terms that have been used to describe a concept in different time periods. Kibana [5] is an open-source interface for retrieval and visualization of data that we use to support a user in finding relevant articles from historical newspapers and visualizing basic statistics and comparisons of relevant articles over time. We also provide a novel formulation and implementation of Cross-Context Lexical Analysis (first demonstrated by [31]), which allows users to quantify and visualize changes in word meaning and usage over time. We combine all these tools in a novel system that uniquely involves humans in the loop throughout all text analyses.

3 SYSTEM DESIGN

In this section, we discuss the design principles, architecture, and capabilities of a novel system that integrates Machine Learning (ML) techniques in computer vision and NLP to support flexible workflows enabling users to combine multiple analysis operators to accommodate diverse application needs.

3.1 Design Principles

We propose the following design principles: 1) Reuse existing software to the maximum possible extent to avoid duplicating work and assess the utility of existing software for our application. 2)

Make the system highly modularized, making it easy to extend and reuse each module. This principle enables easy replacement of any component with a more advanced one with similar functions but much faster speed or more accurate analysis results, which we anticipate may become available in the future. 3) Move the computing environment closer to data wherever possible, through the application of Docker containers, to address the issue of transferring large data collections through the network. 4) Integrate innovative text analysis techniques not available in any existing toolkit and design compatible text analysis operators that users can flexibly combine to generate reproducible customized workflow pipelines and visualizations for specific applications.

3.2 System Architecture

Based on the principles discussed above, our proposed system consists of six main phases/modules: Data Collection, Image Preprocessing, Image Analysis, Text Preprocessing, Text Analysis, and Query and Visualization, with each of these phases/modules potentially having some overlap with their neighbors. Each module can have one or more sub-modules that are specific pieces of software that perform specific tasks—for example, a specific kind of newspaper article segmentation algorithm can be a sub-module within the Image Analysis module. Various combinations of the sub-modules will be needed to process a newspaper collection. A central admin web application could be used to turn these sub-modules on and off and configure them with specific parameters. Internally, the orchestration of the modules and the sub-modules will be through the interaction of Digital Transformation, Data Management, and Workflow Management software. There are some excellent candidates for Workflow Management software like Apache Airavata [29], Apache AirFlow [1], Parsl [15], for Data Management like the Clowder Framework [28], and for Digital Transformation like Brown Dog [34, 41]. We will be looking into these and other relevant software before finalizing the design of the system. Our goal is to leverage existing software as much as possible. Figure 1 shows a high-level architecture diagram of the proposed system.

3.3 Text Analysis Support

To enable a single system to support flexible text analytics on historical newspapers that can be adapted to support different social science research questions, we have designed the architecture of the text analysis subsystem to reflect the logical text analysis workflow, leading to three main modules with multiple sub-modules: Text Preprocessing (e.g., spelling correction, stemming), Text Analysis (e.g., word embeddings, semantic search), and Queries & Visualizations (e.g., term frequency histograms), as shown in Figure 2. Each sub-module is implemented as a microservice using Flask.[1] We chose the microservices architecture because it enables us to develop each (sub-)module independently. Depending on the application needs, we can trade accuracy for efficiency or vice versa. More importantly, all the sub-modules can be flexibly combined to create specialized workflows for specific tasks. Section 4.5 presents examples of such workflows. Commonly used workflows could also be recommended to other similar users to assist them with performing similar tasks. This design is analogous to combining and ordering operators (cf. sub-modules) to create functions or "macros" (cf. workflows) in Microsoft Excel. Common macros could be recommended to users to easily apply the selected sequence of operators to their text data.

[1] https://flask.palletsprojects.com/en/2.0.x/

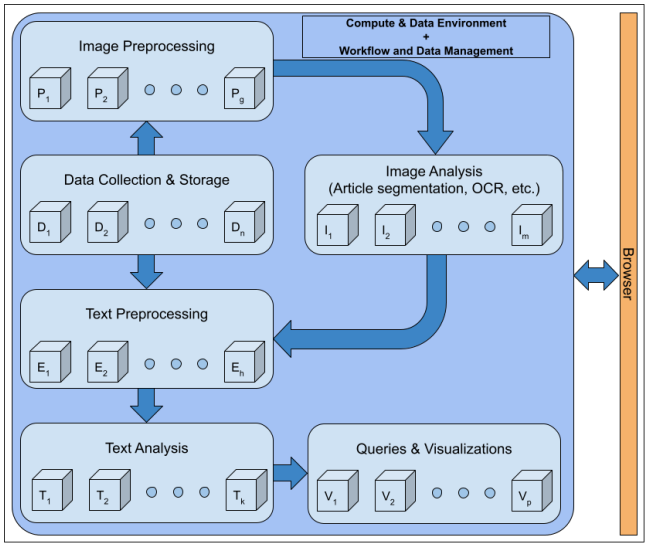

Figure 1: A high-level architecture diagram of the proposed system for historical newspaper analysis showing its various modules.

Another novel feature is that the Text Analytics Support follows a human-in-the-loop model to optimize user-system collaboration in an analysis workflow. Many NLP algorithms are based on supervised ML, whose accuracy depends on the availability of training data. With human-in-the-loop systems, users can easily annotate increasingly more training data to continuously improve the accuracy of a module as needed. The web-based platform developed using Python, Flask and JavaScript engages users in the entire analytics pipeline, including article selection and uploading for further analysis, data annotation to create benchmark datasets for supervised text classification and ML, interpretation of analysis results, and provision of feedback for model improvement. User actions can be recorded to generate log data, which further enables the use of ML algorithms (e.g., collaborative filtering) to recommend effective analysis actions to future users.

4 DESIGN AND IMPLEMENTATION OF A PRELIMINARY SYSTEM

In this section, we describe how we have followed the design principles and architecture in the previous section to implement a preliminary system based on the state-of-the-art AI techniques and software and discuss our observations about their effectiveness. The system consists of six modules.

4.1 Data Collection

This module collects data from relevant sources and downloads it into suitable data and compute resources. As with many big data systems, moving data from one system to another can be costly or time-consuming. Hence, we propose the approach of moving compute closer to data wherever possible. The data obtained from

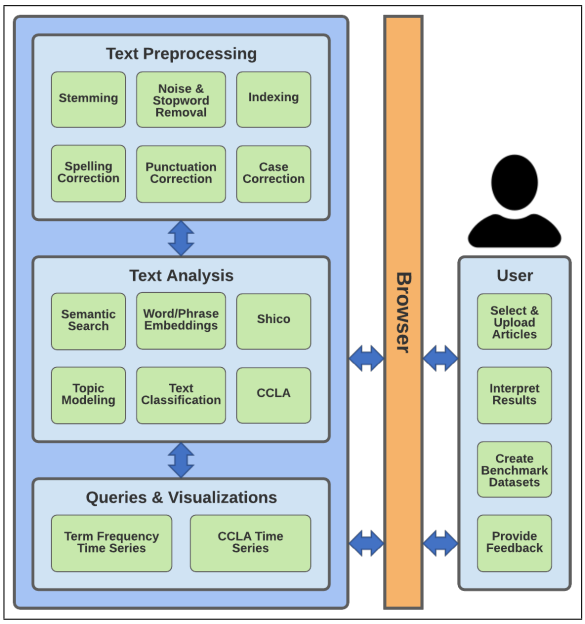

Figure 2: Architecture diagram of the human-in-the-loop Text Analysis Support.[2]

various sources can be heterogeneous in structure, and hence sometimes data transformations will need to be applied to convert data into a processable format.

4.1.1 Library of Congress Chronicling America Newspaper Collection. We started the data collection phase by looking at some free and publicly available historical newspaper collections in the United States. LOC-CA, a well-known newspaper collection, provides access to information about historical American newspapers and select digitized newspaper pages [2]. The digitized newspaper collection consists of JPEG/PDF files containing scanned newspaper pages, XML files containing machine-readable text and metadata encoded in Analyzed Layout and Text Object (ALTO) format, and text files containing plain text obtained after performing OCR. This website provides access to historical newspapers that are in the public domain from about 1777 to 1963. At the time of writing this paper, about 18.8 million pages of digitized U.S. newspapers are available through this website.

Since our work focused on studying the evolution of juvenile delinquency in the U.S. as a social construction, we started by running search queries against the LOC-CA website and downloading relevant output files (JP2, XML, and TXT). Some of these search queries used the keywords *juvenile delinquency*, *delinquency child*, and *delinquency youth*. Later, we designed and developed a command-line application written in Python to perform these searches using the LOC-CA website's Application Programming Interface (API) and perform the bulk download of various data (images, text files, and JSON format metadata) as needed [38]. Using this program, we downloaded 18.4 million text files and 500,000 scanned newspaper images from the LOC-CA collection for the initial analysis.

[2]Here, CCLA stands for Cross-Context Lexical Analysis (see Section 4.5.2).

4.1.2 ProQuest Historical Newspapers™. Later, we got access to the ProQuest Historical Newspapers™ (PHN), another historical newspaper collection containing the text of segmented newspaper articles [7]. From this newspaper collection, we are using about 58 million documents for the initial analysis.

Because the PHN license has restrictions on what we can and cannot do with the data, we had to take steps to ensure we met with all aspects of the licensing agreement.

4.2 Image Preprocessing

The Image Preprocessing module addresses the need for preparing the digitized historical newspapers for the image analysis phase. As with any other computer vision pipeline, image preprocessing is an important step. Some of the commonly used preprocessing steps are image resizing, brightness and contrast corrections, noise removal, sharpening, and geometric transformations. Each newspaper collection may need a mix and match of these preprocessing steps depending on the age, condition of the original newspaper, quality of the scanning process, and image compression method. We can use this module to select the kind of preprocessing needed and set the various parameters needed to improve the images or standardize them before further processing.

4.3 Image Analysis

The Image Analysis module encompasses all the analyses done on the images after the preprocessing step. These can be algorithms used to perform OCR or newspaper segmentation or any other operation performed on the image to retrieve some useful information from it. Here also, there would be many image analysis methods applicable for various newspaper collections. A user would choose those methods relevant to the newspaper collection they are using.

As part of the Image Analysis module, we did a preliminary analysis on the initial set of query results from the LOC-CA website for data quality. We developed a program that overlays numbered rectangles (representing text boxes) on top of the scanned newspaper page after parsing the ALTO XML data [37]. During this initial analysis, we observed some of the limitations of the text obtained from the LOC-CA website. For example, the extracted text was not segmented into individual newspaper articles. It did not always follow the reading order, and the text quality needed to be improved [42]. We later decided to work with the existing data from LOC-CA and improve it by performing newspaper article segmentation and doing re-OCR.

For doing re-OCR, we looked into popular open-source OCR engines like Tesseract [43]. It comes with many options for improving image quality, thereby improving the overall quality of the text output. We applied some of the methods described in its documentation to improve the text obtained from the LOC-CA website [4]. The Tesseract output text file followed the reading order, but it did not do newspaper article segmentation. We also looked into other OCR solutions like Google Cloud Vision API [8]. We created a small Python-based client using the Google Cloud Client Python Library [11] to submit sample images to the Google Vision API for OCR. The results obtained from using Google Cloud Vision API were better compared against Tesseract's output, but it also does not take care of newspaper article segmentation.

For performing newspaper article segmentation, we started by exploring a recent work on newspaper segmentation that uses Deep Learning [32]. Due to its limitations around the unavailability of published training data or trained models, we could not proceed much further. We then started creating a proof-of-concept model for segmenting newspaper articles using Mask R-CNN [12]. This is a well-known open-source implementation of the Deep Learning instance segmentation (the task of identifying object outlines at the pixel level) model by the same name [19]. We forked this repository and added code for segmenting scanned newspaper images [39]. We obtained 44 scanned digital historical images from the LOC-CA website by running search queries like"juvenile delinquency" and "delinquency child". We used these search queries to get images representative of the data used in the research use case driving the system development. From within the search results, these images were randomly selected. We then manually annotated the images using these six classes: ad (advertisement), article, the masthead, run head, photo, banner, resulting in a total of 1129 ground truth annotations. We split these into a training dataset containing 34 images and a validation dataset containing 10 images. We trained the Mask R-CNN model using transfer learning based on a ResNet-101 backbone network [20] that was pre-trained on the MS COCO dataset [26]. For doing transfer learning, we trained only the network heads following similar examples in the Mask R-CNN source code [12]. We also used data augmentation (selected as up to three of these transformations: flipping the image horizontally, rotating the image by +/- 5 degrees, adding brightness to pixels, blurring image pixels) to add more variations to the training dataset. As shown in Figure 3, the initial results of performing instance segmentation on a minimal dataset of historical newspapers show some promising results. However, in some instances, the segmentation is off by a lot. We believe that the model could be improved by adding more training data and training all the network layers. There is a similar work that poses newspaper article segmentation as an instance segmentation problem and uses the same Mask R-CNN model that we also used [14]. The results obtained by their model, which was trained on a larger dataset, look really promising, though it is not clear if it would perform similarly with digitized historical newspapers, which usually contain more noise than their modern counterparts.

Figure 3: Initial results of training a Mask R-CNN model to perform instance segmentation of various regions of the newspaper page[3], including articles.

[3]Image source: Library of Congress, Chronicling America: Historic American Newspapers site; multiple images.

4.4 Text Preprocessing

The text preprocessing module takes in text data from the data collection and image analysis stages and prepares it for further analysis. Currently, it parses text data in the JSON and XML formats using Python. It also performs text cleaning, i.e., removing noisy characters, stemming, and lowercasing using Python and analyzers in ElasticSearch [3]. Currently, we have processed the PHN collection described in Section 4.1.2.

Moreover, as described in Section 4.3, there are multiple OCR errors, such as spelling errors (e.g., 'the' -> 'tbe', 'Of' -> 'Nf'), missing words, and missing punctuation. Such errors reduce the accuracy of the text analysis modules and hamper readability. To overcome such problems, we experimented with several post-OCR correction approaches [17], but found that existing spelling correction [22] and punctuators [45] worked best for our use-case, primarily because they introduced few additional errors. To further reduce the possibility of introducing new errors, we took a conservative approach of only accepting spelling corrections if they were common words (e.g., 'the') or had a small edit-distance (1 substitution) to the original word.

For efficient search, we indexed the newspapers (title, abstract, publication date, publisher URL, and body) in ElasticSearch.

4.5 Text Analysis

The central question that our current text analysis module aims to investigate is how the meaning and usage of a concept varies over time in a given text collection. For this, we support two kinds of analyses: 1) human-in-the-loop semantic searching of articles about a concept, and 2) tracking changes in these terms' meanings and usages. We describe the two workflows below.

4.5.1 Human-in-the-loop Semantic Searching of Articles About a Concept. One common use-case of the system is to help researchers identify articles about a concept, for which an exact match of the concept terms is usually insufficient. We explored multiple approaches for semantic search as described below:

a) Word2Vec Embeddings: One sub-module trains word embeddings. Since the meaning of terms and concepts undergoes semantic shifts, we divided the article corpus into multiple segments by decades and one model was trained per segment. The vocabulary of the models included phrases identified using the Python Gensim Phrases module,[4] which automatically detects common phrases based on collocation.

Our interactive browser enables users to select a Word2Vec model to find synonyms of terms. Further, it automatically generates queries (i.e., Proximity Search Query) for finding those terms and synonyms using Kibana and ElasticSearch. Users can copy and paste the generated queries in the Kibana search tab in our browser.

b) Semantic Concept Shifts Exploration: Since the terms used to describe a concept undergo semantic shifts, it is useful to track such shifts for semantically finding articles about the concept. For this, we leveraged the ShiCo tool. Given a set of seed words, it shows the most related terms based on the cosine similarities between the word embeddings. It also visualizes how the words similar to the

[4]https://radimrehurek.com/gensim/models/phrases.html

seed words changed over time. The visualization is in the form of word network graphs where the words related to each other are connected by a graph edge.

The tool has two different underlying algorithms to find similar words, Adaptive and Non-adaptive. The Non-adaptive algorithm uses only the set of user-provided seed words to find the most similar words for each decade. The Adaptive algorithm modifies the set of seed words over time in either the "Forward" direction, where seed words are updated from the last decade to the earliest one, or "Backward" to do the opposite. For example, using the "Forward" direction starting from the 1990s, it would first find a set of the most similar words to the user-provided seed words using the 1990s word embedding model. This new set now becomes the seed words for the 1980s. This process continues until the earliest decade (the 1800s).

We also modified the tool to accept multi-word (phrase) queries. Phrase embeddings are created by taking the average of the word embeddings of its constituent words.

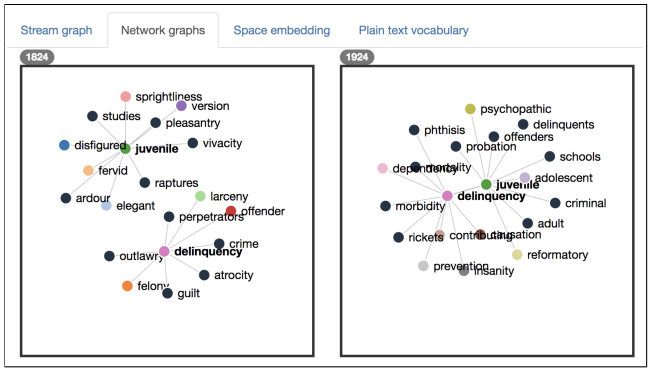

Figure 4: Word Network graphs showing the most similar terms to the input query "juvenile, delinquency" on the ShiCo tool based on Google-Ngram corpus models in 1824 (left) and 1924 (right). Note that there is no association (edge) between "juvenile" and "delinquency" in 1824, but one appears in 1924.

c) Topic Modeling: Similar to word-embeddings, topic modeling is another useful approach to identify related terms and synonyms for semantic search. Thus, one sub-module enables users to run topic modeling (LDA algorithm using Python Gensim) on selected articles (e.g., articles matching a query).

The topics generated by the algorithm can sometimes be hard to interpret. To help with the interpretation, the system shows the top words per topic and topic overviews (i.e., sentences with the highest topic coverage in articles with the highest topic coverage). The number of topics is a meaningful parameter for our envisioned applications, which users can set to different values as needed to examine the mined topics at variable resolutions.

d) Frequency Analyses of Search Terms: Once the terms belonging to a concept are discovered, the next step for a user is to see how the terms were used in the text collection in different decades. For this, we use the Kibana tool because it has features to search and build visualizations on top of large text collections. The search queries in Kibana can be specified in the Kibana Query Language or Lucene syntax.

After initial exploration of Kibana, we found two main visualizations to be useful, Normalized Time Series and Time Series. Normalized Time Series visualizes the fraction of documents (search articles) containing the search terms over time, while the Time Series visualizes the absolute counts of documents. For both the visualizations, filters can be applied based on the time period of interest or other relevant facets of the indexed documents. Further, multiple time series can be shown on the same visualization to compare the time trends for different search queries.

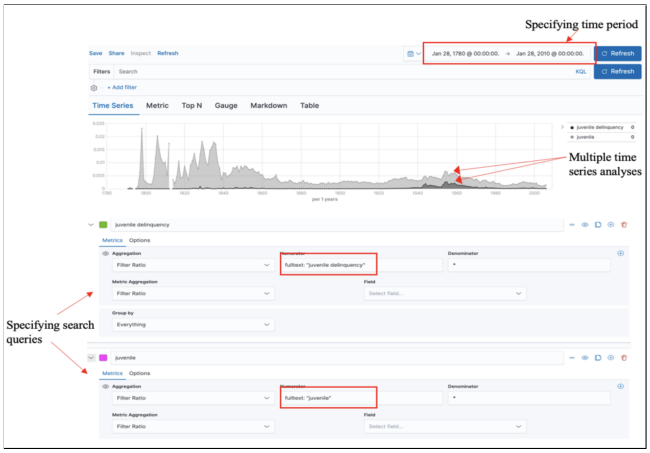

Figure 5: Normalized time-series curves on Kibana showing the fractions of documents matching the search queries over time. The figure compares the fractions of documents containing the terms "juvenile" and "juvenile delinquency" over time.

e) Text Classification with Human-in-the-loop Improvement: In addition to the unsupervised approaches discussed above, the system also enables users to create benchmark datasets by uploading articles and annotating whether an article is about a concept or not. Users can also highlight and save the sentences describing the concept within the articles and take notes. The dataset can be used to train supervised ML algorithms (i.e., Logistic Regression, Random Forest, XGBoost) with TF-IDF features of unigrams and bigrams using Python Sklearn.[5,6]

All the sub-modules mentioned above can be used together for semantic search (see Figure 6). A typical workflow could be to first identify synonyms and terms used to describe the concept (sub-modules *a, b,* and *c*), search candidate articles and explore them using Kibana (sub-module *d*), and find more relevant articles using text classification algorithms that can be continuously improved with human in the loop (sub-module *e*).

[5]https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[6]Currently, unigrams and bigrams are produced with Sklearn's default n-gram tokenizer, but we plan to integrate Gensim Phrases n-grams in this approach.

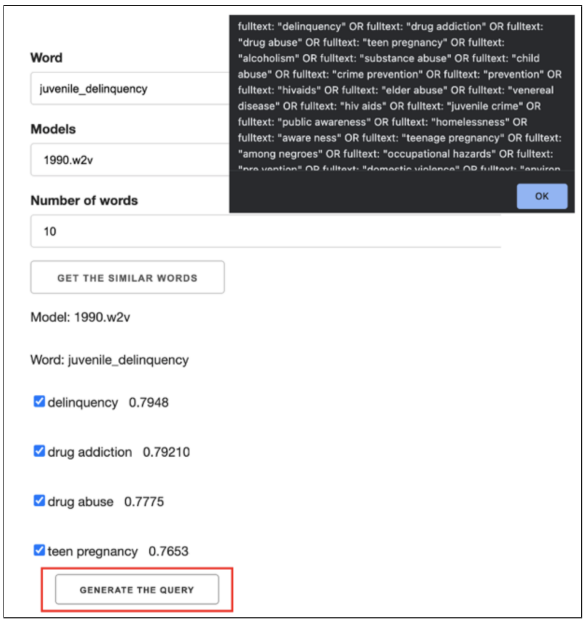

Figure 6: A screenshot of our new tool. Users can search for words and phrases (e.g., "juvenile delinquency'), and the tool returns the most similar words to them based on the selected word embedding model, then generate a Kibana query to find documents containing at least one of these words.

4.5.2 Analyze Changes in Term Meaning and Representation. To analyze how the meaning and usage of terms changed over time, we support the CCLA approach developed by [31], which is not available in any existing text analysis system. In general, it is not possible to directly compare learned representations of word meaning (e.g., embeddings) from independently trained models; but CCLA allows one to perform meaningful comparisons between word embeddings trained on different datasets by computing the similarity of a query word's representation to other word representations from the same model, then comparing these similarities to those of the query in another model.

CCLA can be applied to the data in our system in various ways, depending on the purpose of analysis. Here we illustrate one such example, which we use to perform the experiment highlighted in Section 5.2. We first train Word2Vec models (using the same process described in Section 4.5.1) to create word embeddings for all text data from each decade. For each decade model, given a query, we find the k most similar word embeddings (measured by cosine similarity) in the model to that of the query to form a "similarity vector" of the query with its k-nearest neighbors in the model, where each element of this similarity vector is simply the cosine similarity score between the word and each of its nearest neighbors. Next, we compare the query's similarity vectors between two decades by computing their cosine similarity, yielding a "similarity of similarities" score (CCLA score), which is a single real number in [0, 1] measuring the similarity of the query's representation (relative to its nearest neighbors) in the models for each decade. Finally, scores are normalized for each pair of decades with respect to that of all other tokens (or a random sample of such) at the intersection by subtracting their mean and dividing by their standard deviation.[7] Plotting these scores between successive decades reveals the change in the query's representation.

5 PRELIMINARY EXPERIMENTS

Our system is meant to be a general system to support many different applications in the social sciences, but it has been motivated by a specific research question, namely: how has the concept of juvenile delinquency been socially constructed over time in news media? We use this example to conduct preliminary experiments that demonstrate the usefulness of the current system and illustrate a few remaining challenges.

5.1 A Motivating Example

Juvenile delinquency is a social construction [21]: the category that separates children and adults through the social meanings of variable age boundaries is constructed, as is the notion that they should not be treated in adult criminal court. This social construction laid the foundation for separate treatment of juvenile delinquents. Reformatory and industrial schools were first introduced in Britain in the 1850s for juvenile delinquents. Juvenile courts were not in formal existence until 1906 [33], due to concerns of dangerous juvenile cases [33]. However, it is unclear when and how the term "juvenile delinquency" entered the news media.

The social construction of juvenile delinquency can be examined through newspapers. The news is a venue to understand social reality. People use the news to learn about the social world and make informed decisions. During this process, the reality described by the news is taken by the public even if it does not accord with the underlying reality. Gradually, the news constructs the social reality [13]. For example, some scholars argue that journalists or news organizations may tend to report certain "unusual" news, presenting a reality that is biased to startle, amuse, or excite the reader [18, 27, 35].

In the early 1800s, rising juvenile delinquency had become a big issue in Britain. According to the British Parliament records (1817), 200 boys had been in custody from August 25 1814, to October 1816. In addition, from 1812 to 1816, 4659 delinquent juveniles were transported to Botany Bay, Australia, and 1116 of them were under twenty-one years of age [6]. This trend continued: based on the British Home Office statistics (1846), the proportion of offenders under age 20 increased from 27.3% in 1842 to 30.5% in 1845 [10].

Do these records actually correspond to an increase in deviant behavior of young people? Or are they only an indication that people were more likely to report such behavior to the officials? In other words, did the juvenile deviant behavior become a social problem due to its actual harm or because of social recognition? It will be helpful if we can read through documents related to juvenile delinquency in the past hundreds of years. However, it is impossible to accomplish this task manually. Even the page-flipping of one hundred years of newspaper documents could take months and years. With high-performance computing, our system, for the first

[7] We added this normalization step after discovering that there is considerable variation in CCLA scores by decade, with average scores loosely proportional to the amount of data used to train that decade's Word2Vec model. We found that average word embedding cosine similarities were positively correlated with training dataset sizes. To our knowledge, this effect has not yet been studied.

time, enables addressing this research question using interactive text analysis.

5.2 CCLA

We experimented with the CCLA approach [31] described in Section 4.5.2 to analyze how the meaning and usage of "juvenile delinquency" changed over time (see Figure 7). This analysis was performed with respect to a special aggregator token, "JUDEL", which was created by temporarily replacing all occurrences of a set of terms that have historically been used to refer to "juvenile delinquency" (including, e.g., "juvenile delinquency", "young incorrigible", "teddy boy") with "JUDEL" before training Word2Vec on the PHN data. We obtained this set by asking a domain expert to list several terms that are known to have referred to the concept of "juvenile delinquency" before the term entered popular usage; but in principle, similar lists could be automatically generated using text mining techniques like ShiCo [23, 30] (see Section 4.5.1), optionally allowing users to decide which mined terms are included in the aggregator token. We plan to integrate this workflow in the future.

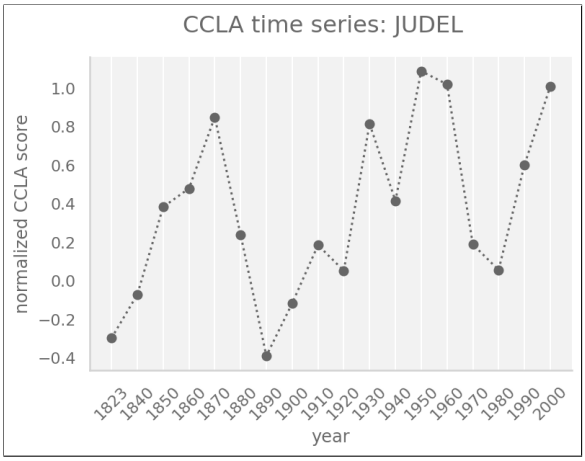

Figure 7: Plot of the (normalized) CCLA score over time with respect to an aggregator token we call "JUDEL" (described in Section 5.2) for k = 11. Each point represents the score computed with respect to the models of the decades to its left and right, where dips correspond to periods in which the query's meaning changed the most.

One interesting trend that emerges in this analysis is the relative semantic stability of JUDEL around 1840-1880, and the rapid change it experiences in the following few decades (see Table 1). Much of this stability is due to its relationship with the words "reformatory" and "reformatories", which quickly rise to the top of the list of JUDEL's nearest-neighbors list in 1840-1850, where they stay until 1890-1910, at which point they rapidly fall down the list. This time period closely corresponds to the adoption and subsequent abandonment of reformatories in the US and UK during this period [40]. Thus, using CCLA, our system is able to highlight periods of rapid semantic change, which users can easily interpret by consulting the nearest neighbors of their query during such periods.

	1840	1850	1860	1870	1880	1890	1900	1910
reformatory	11	1	2	5	2	1	2	11
reformatories	-	2	1	1	1	4	1	4

Table 1: The position of selected tokens in the lists of JUDEL's nearest neighbors.

5.3 Quantitative Evaluation of Semantic Search

Searching for the JUDEL token in Kibana, we found 29 articles out of ~92,000 articles published until 1830. Based on some early exploration, we found that the JUDEL token alone mostly had general synonyms of crime (e.g., "crime", "depravity"), and not youth crime specifically. So, we identified synonyms of both "JUDEL" and "young" in the news articles during 1790-1830 (using word embeddings) and searched for their co-occurrence using Kibana. To ensure higher precision, the co-occurrence was required to be within a set window (10 words).

Using this approach, we identified an additional 237 candidate articles about juvenile delinquency. The domain expert created a benchmark dataset by labeling those candidate articles and found 19 relevant articles (in addition to the 29 relevant articles found using the "JUDEL" token). Thus, the precision of the Word2Vec models was 8% (19/237). While this score appears to be low, it is significantly higher than if we would randomly choose more candidate articles for the expert to annotate, saving a significant amount of time for our expert annotator. Another useful approach using our interactive tool could be that users iteratively refine the query to better capture crime committed *by* young people and not *towards* young people (e.g., using queries like "young boy arrested"), which was a common cause for false positives.

This dataset was also used for training text classification algorithms. The test set accuracy using nested stratified cross validation (or "double-cross" [44]) with $k = 5$ for the outer loop and $k = 3$ for the inner loop is shown in Table 2. Random and Stratified Random are baselines. Random generates predictions uniformly at random and Stratified Random generates predictions by respecting the training set's class distribution. As can be seen, XGBoost had the best performance. This suggests it is feasible to train reasonably accurate classifiers for this task with humans in the loop. The classifier can be continuously improved over time using ML as users annotate more articles, and a user can control the tradeoff between the effort and accuracy based on the need of a specific task. The annotated dataset can be released to the research communities to facilitate new research in ML and NLP.

	Accuracy	Precision	Recall	F1
Random	0.466 ±0.05	0.136 ±0.02	0.38 ±0.12	0.199 ±0.04
Stratified Random	0.688 ±0.05	0.175 ±0.1	0.184 ±0.1	0.176 ±0.09
Logistic Regression	0.842 ±0.04	0.631 ±0.34	0.276 ±0.19	0.354 ±0.23
XGBoost	**0.868 ±0.02**	**0.754 ±0.14**	**0.442 ±0.12**	**0.541 ±0.1**

Table 2: Performance of the classifiers on the test set for predicting articles about "juvenile delinquency."

5.4 Computational Performance and Scalability

Due to the difference in licensing of the data collections we used, we downloaded and processed the data in two different computing environments. For downloading and processing the LOC-CA collection, we used two systems. One was the National Center for Supercomputing Applications (NCSA) Hardware Accelerated Learning (HAL) Cluster [24] and the other one was Pittsburgh Supercomputing Center (PSC) Bridges-2 [46]. We were also supported by the XSEDE Extended Collaborative Support Service (ECSS) program [48]. For indexing, processing, and analyzing the PHN collection, we used an in-house dedicated computing environment called TIMAN.

5.4.1 Downloading Datasets. For downloading the entire LOC-CA collection, we used 1 compute node from PSC Bridges-2 (AMD EPYC™7742 - 128 cores and 256GB RAM per node). We started testing the download program with a few parallel tasks per node and increased that number to 128 in production. We used the Python Multiprocessing package to download different data batches in parallel using the LOC-CA Bulk Data API. We also added a sleep time of 0.01 seconds after each HTTP request to avoid running into any possible rate-limiting issues at LOC-CA. We used about 15,353 core-hours (SUs) for downloading 18.8 million text files, 18.8 million JSON files containing page metadata, 3.5 million ALTO XML files, and 500,000 JP2 images from LOC-CA (and testing the download program).

5.4.2 Image Analysis. For training the Mask-RCNN model, we used 1 compute node from NCSA HAL Cluster (NVIDIA® Tesla® V100 - 1 GPU, 16 CPU cores per node, and 1.2GB RAM per CPU core). The training time was usually around 8-10 hours. The average training image resolution is 5595 x 7653. The images are scaled down to 800 x 1024 during the training process. We understand that the training time can be reduced by using more compute resources. As we add more training data, we will need to improve the efficiency of the training process. Once the model is trained, actually performing the segmentation is almost instantaneous.

5.4.3 Text Analysis. For text analysis modules, we used TIMAN (4 Intel®Xeon®Silver 4210 CPUs, containing 40 cores and 80 threads total).

Indexing PHN took over a week using Python's parallel bulk ElasticSearch indexing function[8] with 4 threads, creating an index of size 175GB.

Training Word2Vec embedding models on each decade of PHN took 47 hours total using 4 cores and 16 worker threads. Once trained, all models take up 17.5GB of space, and may be used to perform a single CCLA query across all decades in ~1.6ms on a single thread. Training the text classifiers took up to 30 minutes, depending on the classifier and its parameters (e.g., number of trees for XGBoost), while testing was essentially instantaneous. With larger training data sets and more sophisticated neural-network based classifiers, GPU-based machines would be more appropriate.

For Topic Modeling, we used Python's parallelized LDA.[9] The processing time mainly depends on the number of documents, iterations/passes, and topics. For example, mining 50 topics from ~7.7 million documents (articles containing the word "juvenile") with 8 threads, 10 passes and 100 iterations took over 2 days.

5.4.4 Potential Challenges. Our system requires substantial offline pre-processing to prepare the datasets (e.g., OCR and text indexing) and training of some analysis modules (e.g., article segmentation and word embedding models). Though these steps can take a long time, they only need to be performed when new data is added or pre-processing modules are updated, which happens relatively infrequently and can be scheduled for periods of low server utilization. Thus, we do not envision that the offline requirements of our system will produce serious performance bottlenecks.

However, the online components of our system may face more significant performance challenges if they need to support many users at once, particularly in storing or processing large quantities of user-generated data (e.g., annotations for human-in-the-loop text classification). Both the ShiCo tool and the Kibana system sometimes took up to 3-4 minutes for complex queries with multiple search phrases over the full corpus. In the future, we plan to improve the performance of the ShiCo tool by incorporating vectorization and parallel processing. Kibana query performance could be improved by increasing and tuning the number of ElasticSearch clusters and the number of shards (currently, we use 1 and 3 of these, respectively). Additionally, while parallel processing of queries in a search engine is mature [16], the best way to support parallel processing of various text analysis operators is an important future research direction whose solutions will most likely depend on the specific operators.

6 CONCLUSION AND FUTURE WORK

In this paper, we studied how to leverage state-of-the-art AI techniques to build a general system to enable researchers to use historical news data to investigate research questions in social science. We presented the design, implementation, and preliminary evaluation of a novel big data analysis system specifically for using historical newspaper collections to perform social science research. The system has built-in pipelines for image analysis and text analysis, accommodates different newspaper collections, and supports interactive text analysis and visualizations. The system was built based on multiple existing software and two new analysis modules that we have developed (i.e., human-in-the-loop text classification and cross-context lexical analysis). We presented some preliminary experiment results using a novel social science research question about juvenile delinquency.

Overall, our experimental results show that a researcher user can already perform meaningful research with our current system by using various text analysis operators supported by the system, demonstrating the great promise of the system as a novel research tool for social science researchers in enabling them to apply many powerful text analysis tools to the large historical news collections and study interesting new questions.

The two new text analysis modules we have implemented have both been shown to be effective. The human-in-the-loop text classifier was able to learn effectively from user annotations to recognize relevant articles with high accuracy, with the potential to continuously improve its accuracy with additional user annotations; and

[8] https://elasticsearch-py.readthedocs.io/en/7.x/helpers.html
[9] https://radimrehurek.com/gensim/models/ldamulticore.html

the CCLA analysis pipeline was able to reveal interesting new insights helpful for research hypothesis exploration and verification, augmenting the intelligence of a social science researcher.

Our results also show that the state of the art software in image processing and text analysis work reasonably well in our application, but there are also many limitations that we still need to address in the future in order to fully realize our vision.

1. Data Preprocessing: We observed the limitations of the text obtained from the LOC-CA website (e.g., lack of newspaper article segmentation and suboptimal OCR text quality), which meant that we could not use this data to achieve the social science research objectives. We also learned that some of the popular software for OCR, like Tesseract and Google Cloud Vision API, does not perform article segmentation well on historical newspapers. Due to these limitations, we are applying image analysis techniques to do newspaper article segmentation and re-OCR.

2. Full integration of multiple modules and operators: Currently, each module in our system operates mostly independently. There could be tighter integration between the individual components. For example, users could be enabled to correct any OCR errors they notice in the results of downstream text analyses, which could be further used as feedback to improve the OCR algorithms.

3. Improvement of text analysis operators: We plan to improve the usability, accuracy and computational performance of individual components of the existing text analysis sub-modules (e.g., add active learning in the human-in-the-loop annotation framework). We also plan to add more types of text analysis functions to make the system more usable by many researchers.

4. Robust analysis against OCR errors: While document-level text analysis methods tend to be fairly resilient to OCR and word segmentation errors (e.g. topic modeling), word-level analyses can be much more sensitive to them (e.g., ShiCo, CCLA), sometimes requiring manual post-processing to generate more reliable results. We need to modify these methods to be more robust to such errors and notify users when manual corrections may be necessary.

5. Trade-off between automation and user control: We generally found a trade-off between system customizability or range of functionalities, which adds to system complexity, and the ease of usage. Future research needs to be done on how to design a system that effectively balances this trade-off. For example, the system could provide scaffolding to users to gradually improve their understanding of the system.

We are currently finalizing the design of the proposed system, connecting different modules together, improving the OCR results, creating new and updating existing operators based on feedback from the community, and running performance tests. We hope that this domain-general system will benefit the community of researchers and enable them to further their research using historical newspaper collections.

ACKNOWLEDGMENTS

This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562. Specifically, it used the Bridges system, which is supported by NSF award number ACI-1445606, at the Pittsburgh Supercomputing Center (PSC) through allocation TG-SES170016. The authors would like to thank the XSEDE Extended Collaborative Support Service (ECSS) program. This work also made use of the Illinois Campus Cluster, a computing resource that is operated by the Illinois Campus Cluster Program (ICCP) in conjunction with the National Center for Supercomputing Applications (NCSA) and which is supported by funds from the University of Illinois at Urbana-Champaign. This work also utilized resources supported by the National Science Foundation's Major Research Instrumentation Program, grant number 1725729, as well as the University of Illinois at Urbana-Champaign. The authors would like to thank the Cline Center for Advanced Social Research and the University of Illinois at Urbana-Champaign for providing access to ProQuest Historical Newspapers™. The authors would also like to thank Benjamin Meier for his correspondence about applying the Mask R-CNN Deep Learning model for newspaper article segmentation.

REFERENCES

[1] [n. d.]. Apache Airflow. https://airflow.apache.org/
[2] [n. d.]. Chronicling America historic American newspapers. https://lccn.loc.gov/2007618519
[3] [n. d.]. Elasticsearch: The Official Distributed Search & Analytics Engine. https://www.elastic.co//elasticsearch
[4] [n. d.]. Improving the quality of the output. https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html
[5] [n. d.]. Kibana: Explore, Visualize, Discover Data. https://www.elastic.co/kibana
[6] [n. d.]. POLICE OF THE MKTROPOLIS. (Hansard, 7 July 1817). https://api.parliament.uk/historic-hansard/commons/1817/jul/07/police-of-the-mktropolis
[7] [n. d.]. ProQuest Historical Newspapers™. https://about.proquest.com/products-services/pq-hist-news.html
[8] [n. d.]. Python Client for Google Cloud Vision — google-cloud-vision documentation. https://googleapis.dev/python/vision/latest/index.html
[9] [n. d.]. The Valley of the Shadow: Two Communities in the American Civil War. https://valley.lib.virginia.edu/
[10] 1846. Criminal Tables for the Year 1845.-England and Wales. *Journal of the Statistical Society of London* 9, 2 (1846), 177–183. https://doi.org/10.2307/2337836 Publisher: [Royal Statistical Society, Wiley].
[11] 2021. googleapis/python-vision. https://github.com/googleapis/python-vision original-date: 2019-12-10T00:10:28Z.
[12] Waleed Abdulla. 2017. Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. https://github.com/matterport/Mask_RCNN
[13] Hanna Adoni and Sherrill Mane. 1984. Media and the social construction of reality: Toward an integration of theory and research. *Communication Research* 11, 3 (1984), 323–340. https://doi.org/10.1177/009365084011003001 Place: US Publisher: Sage Publications.
[14] A. Almutairi and M. Almashan. 2019. Instance Segmentation of Newspaper Elements Using Mask R-CNN. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. 1371–1375. https://doi.org/10.1109/ICMLA.2019.00223
[15] Yadu Babuji, Anna Woodard, Zhuozhao Li, Daniel S. Katz, Ben Clifford, Rohan Kumar, Lukasz Lacinski, Ryan Chard, Justin M. Wozniak, Ian Foster, Michael Wilde, and Kyle Chard. 2019. Parsl: Pervasive Parallel Programming in Python. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '19)*. Association for Computing Machinery, New York, NY, USA, 25–36. https://doi.org/10.1145/3307681.3325400
[16] Stefan Buttcher, Charles LA Clarke, and Gordon V Cormack. 2016. *Information retrieval: Implementing and evaluating search engines*. Mit Press.
[17] Guillaume Chiron, Antoine Doucet, Mickaël Coustaty, and Jean-Philippe Moreux. 2017. ICDAR2017 Competition on Post-OCR Text Correction. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Vol. 01. 1423–1428. https://doi.org/10.1109/ICDAR.2017.232 ISSN: 2379-2140.
[18] Herbert J. Gans. 1979. Deciding what's news: Story suitability. *Soc* 16, 3 (March 1979), 65–77. https://doi.org/10.1007/BF02701600
[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2018. Mask R-CNN. *arXiv:1703.06870 [cs]* (Jan. 2018). http://arxiv.org/abs/1703.06870 arXiv: 1703.06870.
[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. 770–778. https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html

[21] Lars-Christer Hydén. 1993. The social construction of juvenile delinquency: Sailing in cold or hot water. *Young* 1, 3 (1993), 2–10. Publisher: Sage Publications Sage CA: Thousand Oaks, CA.

[22] Sai Muralidhar Jayanthi, Danish Pruthi, and Graham Neubig. 2020. NeuSpell: A Neural Spelling Correction Toolkit. *arXiv:2010.11085 [cs]* (Oct. 2020). http://arxiv.org/abs/2010.11085 arXiv: 2010.11085.

[23] Tom Kenter, Melvin Wevers, Pim Huijnen, and Maarten De Rijke. 2015. Ad hoc monitoring of vocabulary shifts over time. In *Proceedings of the 24th ACM international on conference on information and knowledge management*. 1191–1200.

[24] Volodymyr Kindratenko, Dawei Mu, Yan Zhan, John Maloney, Sayed Hadi Hashemi, Benjamin Rabe, Ke Xu, Roy Campbell, Jian Peng, and William Gropp. 2020. HAL: Computer System for Scalable Deep Learning. In *Practice and Experience in Advanced Research Computing (PEARC '20)*. Association for Computing Machinery, New York, NY, USA, 41–48. https://doi.org/10.1145/3311790.3396649

[25] Benjamin Charles Germain Lee, Jaime Mears, Eileen Jakeway, Meghan Ferriter, Chris Adams, Nathan Yarasavage, Deborah Thomas, Kate Zwaard, and Daniel S. Weld. 2020. The Newspaper Navigator Dataset: Extracting And Analyzing Visual Content from 16 Million Historic Newspaper Pages in Chronicling America. *arXiv:2005.01583 [cs]* (May 2020). http://arxiv.org/abs/2005.01583 arXiv: 2005.01583.

[26] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *Computer Vision – ECCV 2014 (Lecture Notes in Computer Science)*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.). Springer International Publishing, Cham, 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

[27] Walter Lippmann. 1946. *Public Opinion*. Transaction Publishers. Google-Books-ID: YhXLOVc6BsoC.

[28] Luigi Marini, Indira Gutierrez-Polo, Rob Kooper, Sandeep Puthanveetil Satheesan, Maxwell Burnette, Jong Lee, Todd Nicholson, Yan Zhao, and Kenton McHenry. 2018. Clowder: Open Source Data Management for Long Tail Data. In *Proceedings of the Practice and Experience on Advanced Research Computing (PEARC '18)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3219104.3219159

[29] Suresh Marru, Lahiru Gunathilake, Chathura Herath, Patanachai Tangchaisin, Marlon Pierce, Chris Mattmann, Raminder Singh, Thilina Gunarathne, Eran Chinthaka, Ross Gardler, Aleksander Slominski, Ate Douma, Srinath Perera, and Sanjiva Weerawarana. 2011. Apache airavata: a framework for distributed applications and computational workflows. In *Proceedings of the 2011 ACM workshop on Gateway computing environments (GCE '11)*. Association for Computing Machinery, New York, NY, USA, 21–28. https://doi.org/10.1145/2110486.2110490

[30] Carlos Martinez-Ortiz, Tom Kenter, Melvin Wevers, Pim Huijnen, Jaap Verheul, and Joris van Eijnatten. 2016. ShiCo: A Visualization Tool for Shifting Concepts Through Time. In *Proceedings of the 3rd DH Benelux Conference (DH Benelux 2016)*. 1.

[31] Sean Alexander Massung. 2017. *Beyond topic-based representations for text mining*. Ph. D. Dissertation. University of Illinois at Urbana-Champaign.

[32] B. Meier, T. Stadelmann, J. Stampfli, M. Arnold, and M. Cieliebak. 2017. Fully Convolutional Neural Networks for Newspaper Article Segmentation. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Vol. 01. 414–419. https://doi.org/10.1109/ICDAR.2017.75

[33] Tim Newburn. 2002. The contemporary politics of youth crime prevention. In *Youth Justice: Critical Readings*, John Muncie, Gordon Hughes, and Eugene McLaughlin (Eds.). Sage Publications, London, 452–463. Num Pages: 476.

[34] S. Padhy, G. Jansen, J. Alameda, E. Black, L. Diesendruck, M. Dietze, P. Kumar, R. Kooper, J. Lee, R. Liu, R. Marciano, L. Marini, D. Mattson, B. Minsker, C. Navarro, M. Slavenas, W. Sullivan, J. Votava, I. Zharnitsky, and K. McHenry. 2015. Brown Dog: Leveraging everything towards autocuration. In *2015 IEEE International Conference on Big Data (Big Data)*. 493–500. https://doi.org/10.1109/BigData.2015.7363791

[35] Robert E. Park. 1940. News as a Form of Knowledge: A Chapter in the Sociology of Knowledge. *Amer. J. Sociology* 45, 5 (March 1940), 669–686. https://doi.org/10.1086/218445 Publisher: The University of Chicago Press.

[36] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1532–1543. https://doi.org/10.3115/v1/D14-1162

[37] Sandeep Puthanveetil Satheesan. [n. d.]. draw-text-boxes-alto-viz. https://opensource.ncsa.illinois.edu/bitbucket/projects/JUDEL/repos/draw-text-boxes-alto-viz/

[38] Sandeep Puthanveetil Satheesan. [n. d.]. loc-ca-search-download-app. https://opensource.ncsa.illinois.edu/bitbucket/projects/JUDEL/repos/loc-ca-search-download-app/

[39] Sandeep Puthanveetil Satheesan. [n. d.]. sandeep-ps/Mask_RCNN. https://github.com/sandeep-ps/Mask_RCNN

[40] Snell Putney and Gladys J Putney. 1962. Origins of the Reformatory. *The Journal of Criminal Law, Criminology, and Police Science* 53, 4 (1962), 437–445.

[41] Sandeep Puthanveetil Satheesan, Jay Alameda, Shannon Bradley, Michael Dietze, Benjamin Galewsky, Gregory Jansen, Rob Kooper, Praveen Kumar, Jong Lee, Richard Marciano, Luigi Marini, Barbara S. Minsker, Christopher M. Navarro, Arthur Schmidt, Marcus Slavenas, William C. Sullivan, Bing Zhang, Yan Zhao, Inna Zharnitsky, and Kenton McHenry. 2018. Brown Dog: Making the Digital World a Better Place, a Few Files at a Time. In *Proceedings of the Practice and Experience on Advanced Research Computing (PEARC '18)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3219104.3219132

[42] Sandeep Puthanveetil Satheesan, Alan B. Craig, and Yu Zhang. 2019. A Historical Big Data Analysis to Understand the Social Construction of Juvenile Delinquency in the United States. In *2019 15th International Conference on eScience (eScience)*. 636–637. https://doi.org/10.1109/eScience.2019.00094

[43] R. Smith. 2007. An Overview of the Tesseract OCR Engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, Vol. 2. 629–633. https://doi.org/10.1109/ICDAR.2007.4376991 ISSN: 2379-2140.

[44] M. Stone. 1974. Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society: Series B (Methodological)* 36, 2 (1974), 111–133. https://doi.org/10.1111/j.2517-6161.1974.tb00994.x _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.2517-6161.1974.tb00994.x.

[45] Ottokar Tilk and Tanel Alumäe. 2016. Bidirectional Recurrent Neural Network with Attention Mechanism for Punctuation Restoration. In *INTERSPEECH*. https://doi.org/10.21437/Interspeech.2016-1517

[46] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. Scott, and N. Wilkins-Diehr. 2014. XSEDE: Accelerating Scientific Discovery. *Computing in Science & Engineering* 16, 05 (Sept. 2014), 62–74. https://doi.org/10.1109/MCSE.2014.80 Place: Los Alamitos, CA, USA Publisher: IEEE Computer Society.

[47] Ivan Vulić, Edoardo Maria Ponti, Robert Litschko, Goran Glavaš, and Anna Korhonen. 2020. Probing Pretrained Language Models for Lexical Semantics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 7222–7240. https://doi.org/10.18653/v1/2020.emnlp-main.586

[48] Nancy Wilkins-Diehr, Sergiu Sanielevici, Jay Alameda, John Cazes, Lonnie Crosby, Marlon Pierce, and Ralph Roskies. 2016. An overview of the XSEDE extended collaborative support program. In *High Performance Computer Applications - 6th International Conference, ISUM 2015, Revised Selected Papers (Communications in Computer and Information Science, Vol. 595)*. Springer Verlag, Germany, 3–13. https://doi.org/10.1007/978-3-319-32243-8_1

[49] Duo Zhang, Chengxiang Zhai, and Jiawei Han. 2009. Topic cube: Topic modeling for olap on multidimensional text databases. In *Proceedings of the 2009 SIAM International Conference on Data Mining*. SIAM, 1124–1135.

[50] Xiong Zhang, Jonathan Engel, Sara Evensen, Yuliang Li, Çağatay Demiralp, and Wang-Chiew Tan. 2020. Teddy: A System for Interactive Review Analysis. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.