Latent Probabilistic Model of News Sources

Project Alpha Prototype Report

Army Cyber Institute

Machine Learning for Media Bias

William Hiatt, Gabriel Matthew, and Deven Biehler

02/21/2023

TABLE OF CONTENTS

I. Introduction **4**
I.1. Project Introduction **4**
I.2. Background and Related Work **4**
I.3. Project Overview **5**
I.4. Client and Stakeholder Identification and Preferences **6**
II. Team Members - Bios and Project Roles **6**
III. Project Requirements **6**
III.1. Spike Stories **6**
III.1.1 6
III.1.2 6
III.1.3 7
III.1.4 7
III.1.5 7
III.2. Functional Requirements **7**
III.2.1. Sentiment Analysis 7
III.2.2. Topic Modeling 8
III.2.3. Similarity Analysis 8
III.3. Non-Functional Requirements 8
IV. Software Design **9**
IV.1. Architecture Design 9
IV.2. Data Pipeline Design 12
IV.3. User Interface Design 13
V. Test Plan **14**
V.I Functional Test Cases 14
V.II Non-functional Requirements Testing Strategy 22
VI. Staging and Deployment Plans **22**
VII. Alpha Prototype Description **23**
VII.1. Data Collection 23
VII.1.1. Functions and Interfaces Implemented 23
VII.1.2. Preliminary Tests 24
VII.2. Topic Modeling 24
VII.2.1. Functions and Interfaces Implemented 24
VII.2.2. Preliminary Tests 24
VII.3. Sentiment Analysis 24
VII.3.1. Functions and Interfaces Implemented 24
VII.3.2. Preliminary Tests 24
VIII. Alpha Prototype Demonstration **25**
IX. Future Work **26**
VIII.I Second Model–Article Comparison-level Model 26

VIII.II Final Model–Ecosystem-level Model 26
X. Repository Information 27
X.I Repository URL 27
X.II Last Commit 27
X.III Contributors Names and Github Usernames 27
XI. Glossary 27
XII. References 28
XIII. Appendices 29

I. Introduction

This document serves as a guide to the Latent Probabilistic Model of News Sources. It will present new and updated details of the current state of the project as well as the direction the project is going. It will include information such as the social science aspects of the project, the open source models we will be using, models we will be creating, research that we will be using and research we will be doing internally, and any requirements that we currently have. Finally this document will quickly go over the bio's of group members on the team that are working on developing the model as well as all the stakeholders that the project will affect.

I.1. Project Introduction

The US Army Cyber Institute needs a method for evaluating a media landscape that does not rely upon expert opinions, subjective interpretations, or knowledge of internal publishing practices. This model will be used to understand the articles and publishers within a media landscape, based on a statistical measure of their biases of topics within the articles. The model will measure major topics and the bias surrounding them within news articles, and return a visualization of the news landscape regarding the topics and the sources reporting on those topics. With this method the Army will be able to better understand information environments. The end goal of this project is to create a new method for the Army that can take in individual news articles from various sources and produce a model of those news sources, based on their articles, which can then be aggregated to a model of the media ecosystem of those news sources. Given that the major differences between news sources is what they choose to talk about (e.g. topic bias) as well as how they choose to talk about those topics (e.g. framing, word choice bias), the basis of the model will utilize topic and sentiment as its primary means of characterization. The model will establish an ecosystem of news articles and sources that is based on biases and sentiment values based on the major topics within each article. This ecosystem will be displayed graphically for the user to make insights about the data. The model will use a mixture of currently available software as well as software created by the team.

I.2. Background and Related Work

Concerns about media bias and its potential impact on society have grown in recent years. Misinformation, polarization, and even discrimination can result from biased news. The information domain, of which the media landscape is a major component, is a critical domain to the success of modern military operations. This has led to the development of NLP tools and software that can accurately predict various information about news articles, such as political stance and subjectivity of an article.

A public tool that is in line with this project is the media bias detection in AllSides(allsides.com, 2022). Their website, allsides.com, provides free bias ratings on major current events, as well as displays articles from multiple political sides. They use a combination of patented news software with a variety of manual methods, like third-party blind surveys and editorial panel reviews, to display the political stances of each side on a topic. They also include key quotes and context, as well as a link to each article in full. This is important for this project because it shows a publicly available tool that functions in a more manual way than this project's software.

Another similar tool is Media Bias/Fact Check through the website mediabiasfactcheck.com(mediabiasfactcheck.com, 2021), which is a public manual service that shows media bias as well as performs fact-checking on major news sources in their database. They rely on a combination of 3rd-party services as well as professional reviews to produce their results, which means their data is generated mostly manually. They provide an in-depth explanation of the methodologies used to rate the articles' media biases, and how it translates onto a spectrum of bias.

There is a multitude of research being performed on topics similar to this project's focus. For example, in "Machine-Learning media bias"(Samantha D'Alonzo, 2021) published out of M.I.T, they present an automated method of media bias that maps newspapers on a two-dimensional scale of left-right bias and establishment bias. They also map topics within those articles to show how the word choice of topics affects the bias authors have. M.I.T's study is very similar to this project in that they effectively mapped a media landscape regarding the topics and sources of news articles.

Another similar study is "Detecting Media Bias in News Articles using Gaussian Bias Distributions"(Wei-Fan Chen), uses sentence-level bias detection software to feed into a model that produces article-level biases based on the topics covered. They use a term second-order bias information to measure at the article level, which they define as the frequency and position of biased statements regarding a topic. This helps them create a model of the bias within an entire article. This work parallels our project, with the major difference being their reliance upon second-order bias information to produce article-level bias analysis.

I.3. Project Overview

The media has a huge duty in the form of distributing news to the American public. The information plays a huge role in how people vote. The current offerings in characterizing media bias aren't as detailed as they should be and still contain a bias in themselves as it's often a human who is assigning these biases. For example, allsides.com has CNN ranked as far left while adsfontesmedia.com has them as slightly left leaning. This is just a single example, but it shows that there is even a bias within deciding bias. In addition one post from a media outlet could contain no bias and be factually correct while another may contain a lot of bias and not be factually correct; the labels assigned to a media source are in practice overly simplistic and media sources are more probabilistic than deterministic in their displays of bias. Finally, current methods of characterizing bias or media landscape are context-bound; the same methods and measures to assess the bias in media landscapes in the United States may not translate to other media landscapes, like those in other countries or regions of the world. The goal of the team is to combat this by creating a model that can model news sources in a probabilistic way, using recent research into bias and topic modeling in order to create an purely empirically driven way, that does not require human subjective judgements, to characterize a media landscape.

The team, with the help of mentors at the Army Cyber Institute(cyber.army.mil), will create a new method that will address the issues presented in the current media landscape, with a focus on bias . Through research, designing, building, and testing the team will deliver a media landscape assessment tool to help fight information warfare.

The project is being built from the ground up. There are no limitations on how we create the model. The team has decided to use Python to create the model. Python is extremely popular for machine learning and will be a great platform to use. The code will be housed on GitHub, this allows for easy version control and collaboration.

Using Python allows for the use of some popular tools. Some tools the team will be using include, spaCy(spacy.io) for natural language processing, Gensim(radimrehurek.com/gensim) for topic modeling, textBlob(textblob.readthedocs.io) for sentiment analysis, Pandas(pandas.pydata.org) for managing data, and Newspaper3k(newspaper.readthedocs.io) for web scraping. These tools will all be used in conjunction to create a model that fits the needs of the client.

The team is also in contact with two Social Scientists at Army Cyber Institute to help in finding the most important bias's to use as well as the best way to display the output. The team will be using sentiment analysis and fact selection as the primary drivers in deciding a bias. As the project progresses, we plan to include additional secondary drivers.

I.4. Client and Stakeholder Identification and Preferences

Our client is the US Army Cyber Institute(cyber.army.mil) with Senior Research Scientist Iain Cruickshank as our mentor and primary contact for the project. The product will be used and maintained by the Army. There are several stakeholders within the Army Cyber Institute including Iain and his colleagues.

II. Team Members - Bios and Project Roles

William Hiatt is a 4th year software engineering student at Washington State University. His skills include C#, Python, SQL, Java, C++, and GoLang. He has prior experience working as a software engineer intern at Kochava as well as a junior web developer at Washington State University. For this project, William will act as a team lead, main developer and designer of the model, and work with his team to create a viable machine learning model.

Gabriel Sams is a 4th year Computer Science major at Washington State University. His skills include C#, Python, Database development/management, SQL, machine learning, software development, agile process, test-driven development, and data science. His prior experience includes mobile application development, database development and management, and unit, end-to-end, and functional testing. For this project, Gabriel will act as a main developer and designer of the model, and work with his team to create a viable machine learning model.

Deven Biehler is a 4th year Computer Science major at Washington State University. His skills include C/C++, Python, SQL, machine learning, HTML/CSS, agile process, and data science. For this project, Deven will act as a main developer and designer of the model, and work with his team to create a viable machine learning model.

III. Project Requirements

III.1. Spike Stories

III.1.1

Determine the tools to be used for sentiment analysis/textual analysis. Users will be able to view per-article data gathered from NLP and sentiment analysis. The data will be clearly displayed and comprehensible so the user can interpret it in a professional setting. Our team will need to determine open-source tools that will provide the highest accuracy and reliability to our data.

III.1.2

Develop a model at every step of document and ecosystem analysis to build a usable model of media bias detection. The user will be able to gather useful data from three compounding levels of analysis. The tool itself will focus on displaying the "news ecosystem" for users, so the previous layers need to provide useful data to construct it. We will explore and develop algorithms to compile meaningful results at every level of news bias modeling.

III.1.2.1

Create the base model for our software, that takes a single document and provides data on its sentiment of topics within the text as well as factuality of the documents. This model will be used to build a larger corpus of data for many separate documents.

III.1.2.2

Develop a model that compares single documents to each other to find matching topics, and sentiment on a per-article basis. These comparisons will be stored to create a larger ecosystem of article comparisons.

III.1.2.3

Building from the per-article model, create a model that compares and displays the overall ecosystem of articles and how they relate to each other on a certain topic. The model will compare sentiment and factuality based on the models' textual sentiment analysis and group them/sort them in a meaningful way.

III.1.2.4

The final model must be able to categorize and sort the articles to show meaningful relationships between articles that match topics. Develop an algorithm to meaningfully sort the article ecosystem into a graph that is comprehensible to the users.

III.1.3

Find the most meaningful ways of measuring bias in text. Users will need the most accurate bias data for each topic. By researching and talking with social science professionals the team will create the most effective way at measuring bias so that the users have the most accurate information available. In the beginning only a couple different forms of bias will be evaluated with plans to expand in the future.

III.1.4

Explore methods of fact-checking as well as opinion recognition. Users will need to see when an article is falsifying data or otherwise lying to promote a bias. Fact-checking is an essential point of data when determining the motives of a particular article or source. We want to research possible ways of assessing the validity of a fact. Fact-checking will allow us to determine if a piece tries to persuade a user into a specific worldview. If we can find the lies, we can detect future lies that could be told in articles yet to be published.

III.1.5

Determine tools to be used for topic modeling. In order to determine the bias per topic, we need a model to extract the topics from the articles. The topics will be extracted from each article and displayed each with a range of sentiment, ranging from positive or negative toward the topic.

III.2. Functional Requirements

III.2.1. Sentiment Analysis

Per Topic: Using a pre-trained natural language processing model, we receive an emotionality score towards an array of topics. Sentiment analysis is crucial to predict the emotional bias towards a specific topic.

Source: Senior Program Manager with Army Cyber Institute originated this requirement. The requirement is necessary for the bias analysis.
Priority: Priority Level 2: Essential and required functionality

Per Article: We receive an emotionality score towards the entire article using a pre-trained natural language processing model. The goal is to give the model more crucial data about the article's topic.
Source: Senior Program Manager with Army Cyber Institute originated this requirement. The requirement is necessary for the bias analysis.
Priority: Priority Level 2: Essential and required functionality

Per Source: Using a custom-built machine learning model, we can determine where the source will lay in the entire ecosystem of media news sources. The goal is to allow a user to understand the worldview of a source within the media ecosystem.
Source: Senior Program Manager with Army Cyber Institute originated this requirement. The requirement is necessary for the bias analysis.
Priority: Priority Level 2: Essential and required functionality

III.2.2. Topic Modeling

Modeling Relevant Topics: Determining the topics is essential to model a per-topic bias analysis. The model learns bias towards an array of topics to portray the media ecosystem to the end user.
Source: Senior Program Manager with Army Cyber Institute originated this requirement. The requirement is necessary for the bias analysis.
Priority: Priority Level 2: Essential and required functionality

III.2.3. Similarity Analysis

Similar Article Analysis: When learning the bias of specific sources, we can use their similarities to other sources to predict missing values. The model will look for any blatant copying from sources and take it as valuable data to make accurate predictions.
Source: Senior Program Manager with Army Cyber Institute originated this requirement. The requirement is necessary for the bias analysis.
Priority: Priority Level 1: Required for an accurate model

III.3. Non-Functional Requirements

Reliability/Accuracy

This tool should most importantly provide accurate and reliable information based on the input given. The tool should provide the same results for the same input every time, as well as provide data that is usable in a professional news media environment.

Extensibility

This tool will be extensible to other domains within online media, including other languages and other political environments. While the tool itself will only work for English language and American news sources, it will allow for the insertion of new tools and libraries to adapt it to these new domains as its users see fit.

Usability

This tool will aim to be usable by industry professionals and those who have domain knowledge of American news, but not necessarily any knowledge of Computer Science or how the tool functions in terms of programming. This means that the tool will include an instruction manual or other documentation that helps users work with the tool. The tool will be designed in an intuitive way.

Content-Focused

This tool will focus on analyzing the actual content of news articles rather than the context surrounding them. The tool will analyze the text itself as well as relevant data and sources to make its calculations–it will not focus on the author, the timing of the article, or other contexts involving the news piece being analyzed.

Comprehension

This tool will provide highly comprehensible data that can be interpreted by industry professionals to make professional decisions in a business setting. It will provide any relevant data to the user and will document any unreliable data or data that could be up to interpretation. Our goal is to be as transparent as possible about the reliability and usability of the data.

IV. Software Design

IV.1. Architecture Design

IV.1.1. Overview

Our project revolving around a latent probabilistic model of online news articles utilizes an architecture of three models. Each model builds upon the previous model, and together creates a finalized product. Our team decided on the 3-model approach due to the fact that our work is presented in multiple stages: article-level sentiment analysis and data gathering, article corpus-level topic modeling and localized sentiment analysis on topics, and sorting/filtering as well as data visualization. These three models can function separately from each other (given the correct data input) but together feed data through each stage which at the end provides meaningful insight into the article landscape it was provided. Here we provide an overview of each component in the diagram provided below, with details provided in the following sections. The web scraping component is the first step in our pipeline, that takes in a list of web article URLs and scrapes the article text from each of them, storing it inside a data structure in local memory. This component also filters out unwanted articles (any that give errors or are inaccessible,) and parses the text by sentence to better read it in. We then pass the web-scraped data through to our topic modeling component. In the topic modeling component, it takes the entire corpus of texts that we have stored from the previous components and applies topic modeling functions to them. This creates a topic space, in which all of the individual articles exist and relate/differ from each other based on topic words and topic prevalence within the articles. That data is then passed through to our sentiment analysis and parsing components, which calculates sentiment analysis on the text at various levels. It performs sentiment analysis on the entire article, as well as on localized portions of the article for individual topics to get a sentiment score for a topic within an article. Finally, after topic modeling and sentiment analysis has been performed, the processed data is moved into the visualization and UI component. This component produces interactable plots for the user to make assumptions on the topic space and sentiment around those topics. The user can filter and search through the data to find specific topics, specific articles, and their associated sentiment analysis values. Extensive detail on these components can be found in the sections below. Directly below is a diagram of our system architecture layout.

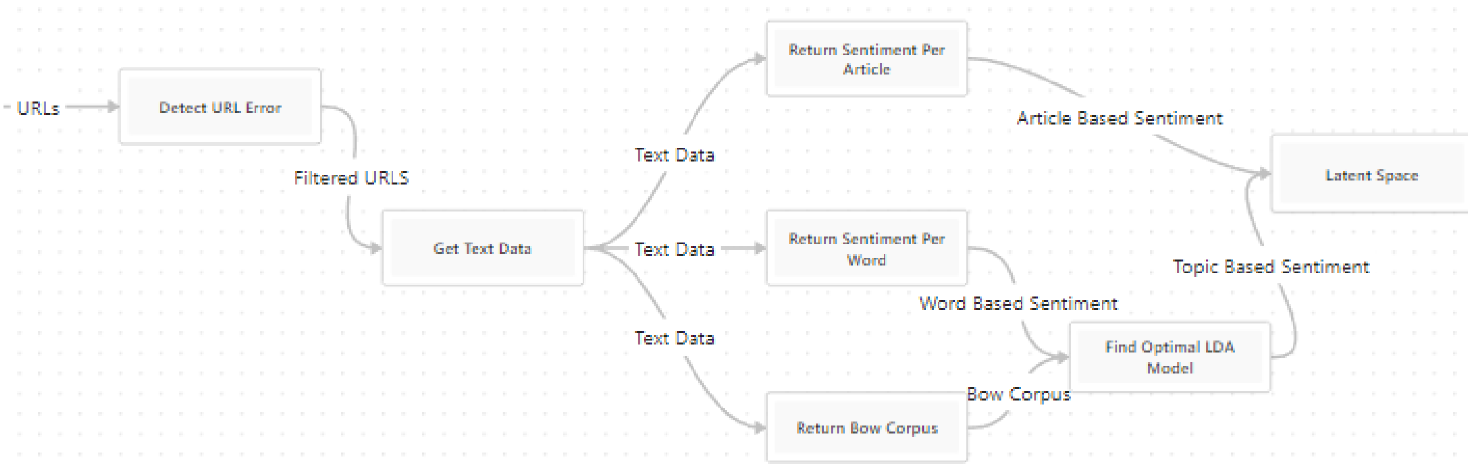

Diagram IV.1 Software Architecture Overview

IV.1.2. Model Input

The model will take in URLs of news articles in the form of a list, and will need to scrape the data from each URL to gather the title, author, and text of the article. The specific input will be a single news article that the model will run on. We think this is the best choice of data input due to a URL being simple to gather and run in the model, and that analyzing a single article at a time will make it easier to compile the data for many articles.

IV.1.3. Data Scraping/Cleaning

The model first reads in each URL from a CSV file and stores them in a dictionary. Each URL in the list is then run through Newspaper3k (https://newspaper.readthedocs.io) to scrape the web page. The gathered text is placed alongside each URL within the dictionary. Newspaper3k was chosen as it was the best web scraping tool for eliminating garbage such as ads, images, and other article links.The library is authored and maintained by Lucas Ou-Yang. Newspaper3k can scrape from 10+ different languages and can even auto detect languages. It extracts from html and can extract things such as text, top image, and all other images in the article. Once it's extracted it can then go through the text and extract information such as keywords, summary, author, and google trending terms. Newspaper3k also uses a lot of python-goose's parsing code. Python-Goose(github.com/grangier/python-goose) is a library that extracts information such as main text from an article, main image of an article, article descriptions, and more. Python-Goose was rewritten by Xavier Grangier.

The text is then stripped of any white spaces. Then the data is loaded into SpaCy(spacy.io). SpaCy then uses TextBlob(textblob.readthedocs.io) to run the sentiment analysis. After it has looped through every URL it then is saved into a dictionary with columns for Sentiment Score, Sentiment Label, Positive Words and Negative Words.

This process allows for easy usability as the user only needs to input the URLs of the news articles without having to worry about scraping the data themselves.

IV.1.4. Topic Modeling Analysis

We will use Gensim(radimrehurek.com/gensim), a Python package, for topic analysis. Our model needs to extract topics from each article to be able to categorize it within our media ecosystem. We picked Gensim because of its efficient implementation, and because it provides the LDA(Latent Dirichlet allocation) topic model algorithm. LDA has the ability to assign multiple topics to a single document, along with a relevance score for each topic on the article. LDA can also be customized with various parameters to optimize for particular use cases. This adaptability can be beneficial in detecting news bias, where various topics or biases may necessitate different modeling strategies.

IV.1.5. Sentiment Analysis

We will be using SpaCy(spacy.io) as our Sentiment analysis tool, as well as the TextBlob(textblob.readthedocs.io) library within Spacy. The model will take in the article's text and perform an overall sentiment analysis on the article and return a polarity sentiment score (it also returns a 'subjectivity' score and keywords related to the scores). We then give the sentiment a positive or negative label, positive for scores over 0 and negative for scores under. It will also perform localized sentiment analysis on topics to find the sentiment around certain topics within the article. This will be necessary to feed our model the topics and sentiment analysis per article to create a larger ecosystem of articles. We chose these tools because of their ease of use, and public access.

Textblob is particularly useful for our model's sentiment analysis due to its flexibility of accessing sentiment, subjectivity, POS tagging, named entity recognition, and various methods of finding each of those within a block of text. It is also already available within the SpaCy package that we are using so it works well with other libraries in the project. The data structures within SpaCy are easily translatable into TextBlob data structures for sentiment analysis. TextBlob also has multiple internal algorithms for sentiment analysis that are highly customizable to fine-tune the model.

IV.1.6. Visualization

A visualization of the media ecosystem will be created as the model output.The first visual option would be a graph on the entire topic landscape and sentiment analysis. This will show all articles and their personal topic word banks, as well as their overall sentiment analysis scores per-article. This is presented in the form of a 2-dimensional bubble chart that is also interactive (explained in the interactivity section.)

From there the visuals can be filtered down to provide more detailed information. One option is to view the article landscape by topic. Through selection of a topic word bank, the user will be able to view the articles with high matches in the topic. This will show the user articles that all share the same topic, and will display their localized sentiment analysis for those topics. This will create a small-scale article ecosystem based on the articles that all share a topic, allowing for more insight into a specific topic. There will also be links between articles that show similarity in text and sentiment. This will provide a comprehensive and understandable format of the data for our users. We chose this solution approach because the model needs to be understandable by industry professionals who don't program.

IV.2. Data Pipeline Design

IV.2.1. Incoming Data

The data coming into the model will be a CSV of news articles on a specific news event. The CSV will have a column of URLs to the different news articles. From this CSV the model will extract all the text needed from the articles. The URLs can include sites such as youtube and twitter however the model will exclude these from the analysis.

Method Name:	Method Input:	Method Description:
scrapeData	CSV file of URLs	Method used for scraping the data. The data won't be scraped if the URL is in the whitelist. If there are any errors in the data scraping an error message will be returned instead of the text. If the URL is not on the whitelist and didn't produce any error the scraped text will be returned.
sentenceLevel	The parsed text	The method will break the parsed text into a list of all sentences within the text. After it is broken apart a list will be returned of all the sentences.

Table IV.2.1 Pipeline Methods

IV.2.2. Data processing

Once the data is fed into the model the URLs are scrapped using Newspaper3k, an article scraping and curation tool developed by Lucas Ou-Yang.

IV.2.2.1. Pre-Scrape check

The main text from the article is scraped if the URL doesn't contain one of the white listed words such as youtube.com or twitter.com. This is done because these social media sites often only have a video as the substance of the topic. These videos can't be scraped so the text that is scraped is usually from the comments on the video or tweet.

URL:	Reason:
https://www.youtube.com	The videos can't be scraped so the description and comments are scraped instead.
https://youtu.be	Some videos are formatted with this URL instead of youtube.com, both

	have the same drawbacks.
https://www.facebook.com	We aren't interested in this data as these aren't credible sources writing these posts.
https://twitter.com	The people tweeting aren't always credible resources and often a video is all that is posted resulting in the comments being what is scraped.

Table IV.2.2.1 Whitelisted URLs

IV.2.2.2. Scraping

Newspaper3k scrapes the entire article. From that entire article we use the body of the article. Using just the body allows us to avoid running analysis on things like the title, any ads on the site, and previews of other articles.

IV.2.2.3. Post-Scrape check

After scraping is complete a few conditions are checked to make sure that the data we have is valid data. These checks include a word count and repeated phrases. The word count is used to avoid any articles in which the HTML doesn't allow for proper scraping. One example is sometimes an image at the top of an article is scraped instead of the entire article, since just the image is scraped it will have a low word count and we can discard it. Checking for repeated phrases allows us to avoid running analysis on articles that have scraping blockers in which the scraper returns something such as "Denied Access, Denied Access, Denied Access,...", running any analysis on this data would be a waste of resources and thus these articles are removed.

IV.2.3. Data management

Once the data has been collected we do a few different things with it. First the data from all the different articles are added into a single string. This allows the user to do analysis on the entire data set as a whole. Second it is run through a method that splits the article into a list of all the sentences within an article. Lastly, the data is sent into the different tools used for analysis. Since the model is modular it allows the user to select which analysis tools they want the data to be run through. After the analysis is done on the data it is then sent to be reformatted and then displayed to the user.

IV.3. User Interface Design

Our user interface design interacts specifically with our data visualization component described in the Architecture Design Section, and thus has one component: interactivity. We have very specific ways of visualizing the processed data from our pipeline, and we have designed the data visualization to best fit our project. The user will have limited capabilities to adjust and change these visuals based on their needs from the software.

Interactivity
Description: The interactivity component will handle the user input on the data visualization, which is the only part of the project (other than the data input) that the user can interact with. The purpose of this component is to allow the user to generate more specific visuals based on the information they need out of our model.

Concept and Use: The interactive portion of our project will be located inside of our visualization component. All interaction is done with the data visuals. Users will be able to view the processed data for any article or generated topic. This means that a user can click on an article in the space and see sentiment analysis of the article, as well as the topic word bank that corresponds to the article's text. This will vary slightly based on the visual model that the user is interacting with. The overall view of the article space will provide an article's personal word bank, and the overall sentiment score for the article. If split by topic, the model will instead show the article's topic word bank and how it relates to the selected topic, as well as localized sentiment analysis on those topic words.

The user will be able to filter and hide/display certain articles based on the topics generated. Users will be able to select from a menu the specific topic that they want to view articles for, and the data will be visualized to show articles for that specific topic (see Visualization section in Architecture Design). The user can switch between specific topics and the overall article space at will.

V. Test Plan

The main test objective for our model would be to perform accurate comparisons between the gathering of data from an article in the real world, either by a tester making meaningful insights into an article and the ecosystem surrounding it or by a previously annotated set of documents, and the results that our model produces. This can be performed at the article, source, and ecosystem level in order to fully test each step of the completed model. This method of testing would mostly require human input along with the data being looked at, without extra tools. This comparison would show how the performance of our automatic model compares to real world results that we expect it to produce. Below are the primary test cases we will use to evaluate functional performance of the model:

V.I Functional Test Cases

Test Case ID: 1.1
Test Case Name: Single Article Sentiment Analysis
System: Article-level Model
Short Description: Performs per-topic and article-level sentiment analysis on a single article
Design Date: 4/20/23

Pre-conditions

The article must be of a valid input type. Article is provided via a URL link.

Step	Action	Expected System Response	Pass/Fail and Comments
1	Input the article into the Article-level model	Model performs sentiment analysis on the topics created by topic modeling	
2	N/A	Model performs sentiment analysis on the entire article	
3	N/A	Model passes the sentiment analysis data to the article comparison-level model	

Post-conditions
The sentiment analysis data is passed towards the article comparison-level model. Sentiment analysis data is viewable for both the article itself and the topics within the article.

Test Case ID: 1.2
Test Case Name: Single Article Topic Modeling
System: Article-level Model
Short Description: Performs article-level topic analysis and returns most coherent group of topics
Design Date:4/20/23

Pre-conditions
The article must be of a valid input type. Article is provided via a URL link.

Step	Action	Expected System Response	Pass/Fail and Comments
1	Input the article into the article-level	Model performs topic analysis on the	

	model	article, gathering groups of topic words together into a group of topics and their associated words	
2	N/A	Model analyzes coherence of topics and returns the grouping of topics that make the most sense within the article	
3	N/A	Model passes on the topic modeling data to the article comparison-level model.	

Post-conditions
The topic analysis data is passed on to the article comparison-level model. The data returned is at the highest coherence level of the possible topics within the article. The topic analysis data is viewable at the article level.

Test Case ID: 1.3
Test Case Name: Article-Level Metadata gathering
System: Article-level Model
Short Description: Gathers necessary metadata about an article to use in comparing article and source aggregation
Design Date: 4/20/23

Pre-conditions
Article must be of a valid input type. Article has metadata associated with it that is accessible to the model.

Step	Action	Expected System Response	Pass/Fail and Comments
1	Input the article into the article-level model.	The model gathers date written, author, company, and other	

		relevant metadata associated with the article.	
2	N/A	The model ties this data to the article for use in the next models.	

Post-conditions
The article now has associated metadata that is accessible by the other models. The metadata can be displayed to the user for the articles they look at.

Test Case ID: 1.4
Test Case Name: Article Corpus Analysis
System: Article-level Model
Short Description: Given a list of article URLs, the model performs all functions described in test cases 1.1-1.3 on them successfully
Design Date: 4/20/23

Pre-conditions
The article corpus is formatted in a valid way for the input into the model. Each article is of valid formatting for the model.

Step	Action	Expected System Response	Pass/Fail and Comments
1	Pass in the corpus of articles in the Article-level model.	Model performs actions described in test cases 1.1-1.3 on all of the articles, and returns a collection of article data for each article.	

Post-conditions
All articles are now associated with their respective data and passed to the next model.

Test Case ID: 2.1

Test Case Name: Article Comparison and Topic Clustering
System: Article Comparison-level Model
Short Description: Performs clustering based on the topics between multiple articles
Design Date: 4/20/23

Pre-conditions
The topic modeling data for every article is available to this model from the previous Article-level Model.

Step	Action	Expected System Response	Pass/Fail and Comments
1	Request the comparison-level topic clustering from the model. (Process is also done automatically to pass into the next model)	Model clusters articles based on similarity of topics found in the previous model.	
2	N/A	Model collects the clustering data between articles and passes it onto the ecosystem-level model.	

Post-conditions
Articles are now related to the clustering data that shows their relationship to other articles with similar or same topics. The ecosystem-level model now has access to similarity metrics and the clustering data between articles.

Test Case ID: 2.2
Test Case Name: Source Aggregation
System: Article Comparison-level Model
Short Description: Finds articles of the same source and collects them together.
Design Date: 4/20/23

Pre-conditions
Articles have all passed through the Article-level model and have associated metadata included with them.

Step	Action	Expected System Response	Pass/Fail and Comments
1	N/A (Process is done automatically to pass in to the next model)	Model collects articles that have the same author or source and marks them as from the same source.	
2	N/A	Model collects a group of sources that have multiple articles representing that source.	

Post-conditions
Data is now available for each source within the model and their associated articles.

Test Case ID: 2.3
Test Case Name: Sentiment Comparison
System: Article Comparison-level Model
Short Description: Compares sentiment between articles that have similar topics, as well as sentiment between those topics
Design Date: 4/20/23

Pre-conditions
Article corpus has been passed through the article-level model and topic analysis and sentiment analysis data has been collected. Article comparison has been performed on the topic analysis and similar topics have been recorded between articles.

Step	Action	Expected System Response	Pass/Fail and Comments
1	N/A (Process is done automatically by the model)	Model performs sentiment comparison between similar articles and their associated	

		topics, and stores how the articles compare to each other.	

Post-conditions
Articles now have sentiment comparisons between similar articles and topics for use in the final model.

Test Case ID: 3.1
Test Case Name: Ecosystem-level Analysis
System: Ecosystem-level Model
Short Description: Models all articles, their similarities and differences, and their topics as an ecosystem of topics and the sentiment around them
Design Date: 4/20/23

Pre-conditions
Article corpus has been passed through both previous models and has all necessary information to produce an ecosystem-level description of the articles' topics and overall sentiment.

Step	Action	Expected System Response	Pass/Fail and Comments
1	User pushes the article corpus through the previous models and requests an ecosystem-level analysis of either sources or single articles.	Model formulates comparisons between either articles or their sources at the highest level–making comparisons over the entire dataset	
2	N/A	Model sorts the respective data points based on topics, sentiment around those topics, and overall sentiment	
3	N/A	Model creates a graphical	

		representation of the ecosystem of data, highlighting the similarities and differences between the measured values of each data point.	

Post-conditions
The model generates a visual representation of the ecosystem of articles that is understandable to the reader. The model displays information for all data points requested.

Test Case ID: 3.2
Test Case Name: Source/Article Comparison View
System: Ecosystem-level Model
Short Description: Model displays the data gathered between selected sources or articles for the user to evaluate
Design Date: 4/20/23

Pre-conditions
The ecosystem-level analysis is complete and the visual representation of the data is available.

Step	Action	Expected System Response	Pass/Fail and Comments
1	User selects articles or sources that they want to specifically compare.	Model recognizes the selections and gathers the relevant data for each data point, as well as the comparison data between them.	
2	User decides to compare the selected data points by confirming it on the visualization.	Model displays the generated data for each data point, and also displays the comparisons made between them for interpretation.	

Post-conditions
No data is changed on the model. The model displays the data for the user to collect and use. The model resets to the baseline visualization after the data gathering is done by the user.

V.II Non-functional Requirements Testing Strategy

The majority of non-functional requirements of this model revolve around how well it is received by the end user, and the satisfaction of the end user with our product. For NFRs like reliability, usability, and comprehension, our testing will come in the form of end-user acceptance testing. Users will give feedback on our model based on the way they use it in a professional setting, and our team will respond to the needs of the client by modifying our code to best fit the users' needs. For extensibility, content-focus, and accuracy, it mostly relies on the way we build our code and the results of our code design. For this we will perform both the functional testing described above and analyze it based on the metrics our client has given us to determine if it is properly meeting the NFRs.

In terms of performance testing, our team will test the model on varying sizes and complexity of article databases to gauge how it handles increased workload and stress. We will measure system performance/time to execute on various workloads, and we will also measure the success of our functional requirement testing on various workloads to make sure that our model works under varying amounts of stress. The most important form of performance testing for the model will be scalability testing; how it performs under databases of gradually increasing size. By testing with very small to very large databases of articles, we will see where our model bottlenecks or runs into hardware limitations that would have an impact on the end user.

VI. Staging and Deployment Plans

The team will make the code available in a GitHub(github.com) repository so that the client can access and use the model. The document includes a detailed readme that will include everything that is needed to run the code. This includes things such as what libraries need to be downloaded, what format incoming data needs to be in, how to run the model, what the model will output, and lastly how to interpret the data that is output by the model.

Since the model is dependent on some external libraries these libraries will be listed in the readme. The information about what the library does, how it's used in the program, and who made the library will be included. In addition the commands to install these libraries will also be included in the readme so that the client can easily install them all.

The readme will also include how incoming data needs to be formatted. It will also include an example so that users can easily follow the example. In addition a template .CSV file will be included so that the user can easily edit just that file.

A few things will need to be checked before running the model, the readme will cover these things. Step by step instructions will be given so that the client can easily check these things and change them if need be. The document will also include instructions on how to run it in multiple environments such as terminal, VS code, and JupyterNotebook.

The readme will also include what the model is outputting. This will include the different variables that it's outputting along with what these variables are and how they are calculated. This will also include how the raw data is outputted if the client wants just the raw data.

Lastly the document will include how to interpret the data. We have two means of output, a graphical output and a raw data output, we will include how to interpret the data for both these modes of output.

After reading the readme the client should know how to install the model, run the model, and evaluate the data that is being output by the model. The document will be written in a way so that anyone not familiar with the project will be able to read the readme and understand what the model is about and how to use it.

This GitHub repository is all that is needed for the client to use the model. The users of the model are technical professionals so there isn't a need to package the model with any user interface as the client is using the raw model itself. From the GitHub repository the client will be able to download the model locally to start using it. Once the model is downloaded the user needs to include the .CSV file of URLs in the same directory as the model. Lastly the user needs to update the file name in the .py file so that it matches the .CSV that they are trying to run. Once that is done the model can be run, after it's done running it will output the data onto the screen as well as save the data as a .CSV file into the same directory that the model is in.

VII. Alpha Prototype Description

Our model consists of four parts: data collection, topic modeling, sentiment analysis, and topic comparison. These four parts work together to perform analysis on the news articles. Having these parts working together allows the model to perform a more robust and diverse analysis on the articles.

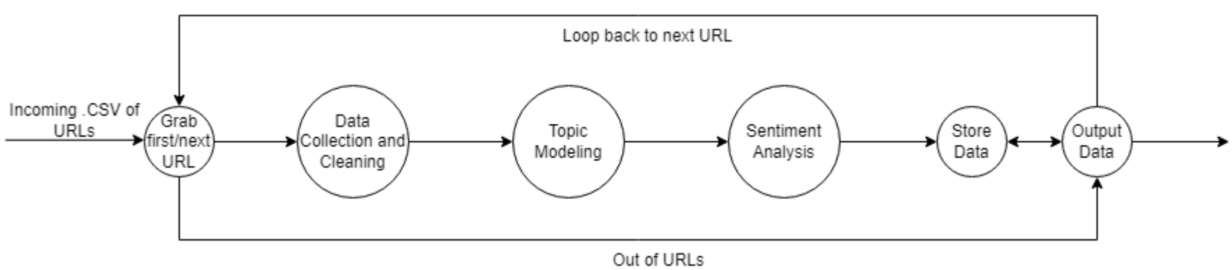

Image 1: Model Pipeline

VII.1. Data Collection

VII.1.1. Functions and Interfaces Implemented

The model takes in a .CSV file filled with URLs. The incoming .CSV file should have one column named “Address”, every row within this column is a URL link to the article that needs to be scrapped. A loop is used to loop through each of these URLs, sending them through the model one by one.

A library named Newspaper3k(newspaper.readthedocs.io) is used to scrape the data from the web. The library is authored and maintained by Lucas Ou-Yang. Newspaper3k can scrape from 10+ different languages and can even auto detect languages. It extracts from html and can extract things such as text, top image, and all other images in the article. Once it's extracted it can then go through the text and extract information such as keywords, summary, author, and google trending terms.

Newspaper3k also uses a lot of python-goose's parsing code. Python-Goose(github.com/grangier/python-goose) is a library that extracts information such as main text from an article, main image of an article, article descriptions, and more. Python-Goose was rewritten by Xavier Grangier.

VII.1.2. Preliminary Tests

Lot's of testing was done to make sure that the data was in the correct format and wasn't adding a lot of garbage. This was primarily done manually, scraping a website and comparing what was scraped vs what was on the website. After testing and adjusting we were able to get rid of a lot of the garbage such as ads, links to other news articles, etc.

VII.2. Topic Modeling

VII.2.1. Functions and Interfaces Implemented

Currently topic modeling is being done on each article. Gensim's(radimrehurek.com/gensim/) topic modeling tool is used to extract the optimal number of topics from a given article. The topics are represented by a list of words from highest to lowest representativeness of any given topic. In the future, topics will be compared with each other to find similar topics that might indicate a meta topic that spans multiple articles. An example of this might be an article about "Quarantine" and an article about "Covid" might be representative of a meta topic called "The Covid Pandemic". This will help the user understand the overall sentiment towards meta topics.

VII.2.2. Preliminary Tests

A Comprehensive score is taken on a wide array of LDA(Pritchard, Jonathan K, 2000) models with a variety of tuned hyper parameters to find the mode comprehensive topic model of a given article.

VII.3. Sentiment Analysis

VII.3.1. Functions and Interfaces Implemented

Currently we are using SpaCy's(spacy.io) sentiment analysis tool to run a comprehensive analysis on all of the articles and topics within the articles. After we receive a list of relevant words from Gensim's(radimrehurek.com/gensim) LDA model(Pritchard, Jonathan K, 2000) we run a sentiment analysis on each word and return a list of sentiment scores. Then we multiply each score by its associated weights given by the LDA model. Adding all scores within a topic together gives us a new sentiment score based on the weighted sentiment of each representative word within the topic. This new score will function as a positive or negative opinion of a topic which can then be compared and contrasted with other articles' topics. We then run a comprehensive sentiment analysis on the entire article for a better understanding of the author's sentiment towards the topic in the title.

VII.3.2. Preliminary Tests

We ran accuracy tests with TextBlob on the BASIL(Zhang, Xinliang, 2022) dataset. We wanted to ensure that Textblob could accurately provide a sentiment score on news data in particular. Sentiment analysis has also been run on a working dataset provided by the client, with a set of 30 and 130 article URLs for basic performance testing.

VIII. Alpha Prototype Demonstration

Problem Statement

A major threat to information security is dissemination of biased media, as it shapes the minds of readers unknowingly towards the author's stance on a topic. If readers are uninformed, they are at the will of the author as to what they will feel about a certain topic, especially if they trust that source to be unbiased. Our probabilistic model aims to give the media analysts and professionals who use it a unique look into the ecosystem of the articles around a certain date and topic. The users will be able to gather meaningful insight into how single articles as well as sources contribute to the overall sentiment of a topic, and analyze trends that emerge from the large interacting body of articles. This model will provide a way for analysts to see how certain sources or articles try to sway their audience, and the repeated patterns of bias that sources use.

The model will take in a corpus of articles on various or the same topic, and give a graphical and data-centric presentation of the factuality and sentiment around certain topics within an article. We can then make comparisons between articles that talk about the same thing, and log their similarities and differences. Finally, the model will take these comparisons and provide an overall view of how articles are similar and different.

Solution approach

Model 1 will be our lowest-level and most complicated tool for building the overall model. It will provide valuable insights into the sentiment of a single news article, as well as around the specific topics included in the article. We will be utilizing free NLP tools including Textblob for localized sentiment analysis, Spacy for web scraping our articles and providing the Textblob library, and Gensim for single-article topic modeling. The model will be able to save data on the entire dataset of articles that are inputted to pass into the next model.

Model 2 will take our analysis of single articles and compare/sort them by topic, as well as providing meaningful insight into the relationships between articles. Model 2 will also group articles by source. The data is being transformed into the relationships between articles in order to pass into the third model, which organizes the articles and displays their relationships, as well as providing the per-article data we create.

The final model is responsible for creating and displaying the ecosystem of sources, including each source individually as well as how they are related to other inputted sources. This will be displayed graphically for users to navigate and create insights out of the data.

Demo

(Sentiment Analysis and article parsing) Here we have the alpha prototype for our article data parsing. The model currently reads in a csv list of article URLs, and translates the URLs into each article's body of text. It also gathers metadata of the article, and we plan to gather the author, title, and date written of the articles if available. With the parsed article data, we run an article-level sentiment analysis on it to get the overall sentiment of the text. We then run a named entity recognizer which gathers all of the named entities (people, organizations, places, dates, times, etc.) from the article text. It then runs localized sentiment analysis of each named entity based on the sentiment of the sentences that the entities are mentioned in. It collects all of the sentiment data around a single entity, and returns the combined sentiment, repeating this process for every entity in the text.

(Topic Modeling) The model also performs topic modeling on the article text, returning a list of the most coherent topics within the article as groups of topic words that together form a single topic. We will use this for per-topic sentiment analysis as well as topic comparison in the second model. By returning a list of the most relevant words, the topic model gives, all together, a sense of what the article is talking about. We are able to visualize the relevance of each topic within the article by the frequency of the relevant words.

Future Work

The next large step in this project is to develop the article comparison model, which takes the data from our first model and compares the article data to create similarities or differences between single articles. It will look at the article topics as well as sentiment to measure where articles are discussing similar topics, and how the articles differ in sentiment around the topics.

Finally, the third model will be responsible for the ecosystem-level analysis. This model will sort articles based on the comparisons made in the second model and create an ecosystem of articles and the topics they relate to. The articles will be clustered and displayed to the user such that they can make valuable insights into the bias that these articles have on certain topics. It will also filter articles to give more detailed feedback on subsets of the entire article list, based on what the user wants to see. Overall the third model will be responsible for showing the overall news landscape and how single data points intersect or differ from each other.

In the future, we plan to create a more robust model by implementing article to article topic clustering. This would allow the model to understand that topics between articles may be related to a larger, overarching, unseen topic. We also want to scale our current model to take in more data and to automatically be fed into the model to update the simulated ecosystem to accurately model the modern ecosystem. Ideas for implementing plagiarism detection and/or fake news detection to increase accuracy of the model is also being worked on.

IX. Future Work

VIII.I Second Model–Article Comparison-level Model

Next semester we plan to develop the second-level model that takes the list of articles and their associated data gathered from the first model. The model will take metadata gathered and group together articles of the same author/source company. It will also make comparisons between article topics and group together articles with similar topics. The model will also compare sentiment values between articles with similar topics to get data on how those articles are similar/differ in their sentiments of certain topics.

Our plan is to develop this model next after finishing the first model, by gathering the data provided in the first model and formatting it to be easily accessible by this model. We will then develop the sentiment and topic comparison together because they access the same set of data from every article. The metadata gathering and source aggregation will be developed after this, because it is a different set of data gathered from the articles. This data will then be formatted and passed into the third model to develop the ecosystem of articles and sources.

VIII.II Final Model–Ecosystem-level Model

In the final model, we will take both the per-article data gathered as well as the insights into how articles/sources compare with each other and format it into a final ecosystem of articles. This ecosystem will display the article-level analysis of all articles as well as the similarities and differences between articles and their topics. This ideally will be sortable by topic, source, timeframe, and other metrics that an end user would want to see. The goal for this final model is to take the vast amounts of data generated by the first two and translate it into readable, and usable insights into the articles provided. The model will be displayed graphically for the user and provide visual data points that the user can expand or collapse based on what they want to view.

We will complete this model last for the project. This model requires accurate output from both of the previous models to work, so they need to be fine-tuned before this model can be finalized. Once the model is ready to be built, we will finalize with the client the best way to present the data for industry professionals. This includes how the data will be displayed, what filtering can be performed on the data, and the scale of which the data will be displayed. Then we will develop the tools for sorting and categorizing each part of the ecosystem and putting them on measurement axes. We will develop the display and how the model handles different sizes of datasets. FInally, each model will be reviewed and fine-tuned for best fit to our FRs and NFRs.

X. Repository Information

X.I Repository URL

https://github.com/WSUCapstoneS2023/Media-Bias-Prediction-Model

X.II Last Commit

Date: 9/10/2023
Commit Description: Created modular code
URL:
https://github.com/WSUCapstoneS2023/Media-Bias-Prediction-Model/commit/617a738517d828a61d7bc20644020459490c3b20

X.III Contributors Names and Github Usernames

Gabriel Sams: gabrielsams
Deven Biehler: Deven-Biehler
William Hiatt: William-Hiatt

XI. Glossary

FR – Functional Requirement.

NFR – Non-functional Requirement.

Article-level Model – The first model in our overall project that refers to the process of gathering a single article, and performing both sentiment analysis and topic modeling on that article to pass into the next model. Also gathers metadata relating to author, source, and date written. The model does this for every article in the database provided.

Article Comparison-level Model – The second model and intermediary stage of the final project. This model makes comparisons between separate articles and calculates similarity between articles based on their topics and sentiment around those topics. Also groups articles by source. This model will pass its data on to the final model, the ecosystem-level model.

Ecosystem-level Model – The final model of the project and the model that takes in data from the other two previous models. Given that data, this model compares and sorts both articles and sources based on their similarity from the second model, and displays it in a comprehensible way for the user. The model will also filter the data and show smaller subsections of the dataset to provide the user with smaller-scale comparisons. The idea behind this model is to display the 'ecosystem' of articles by showing how they all compare to each other and fit on a scale of sentiments on their topics.

XII. References

Anderson, Martin. "MIT: Measuring Media Bias in Major News Outlets with Machine Learning." Unite.AI. Unite.AI, December 10, 2022. https://www.unite.ai/mit-measuring-media-bias-in-major-news-outlets-with-machine-learning/.

"Army Cyber Institute (ACI) Home." Army Cyber Institute (ACI) Home. Accessed April 20, 2023. https://cyber.army.mil/.

"Balanced News via Media Bias Ratings for an Unbiased News Perspective." AllSides. AllSides, October 20, 2022. https://www.allsides.com/unbiased-balanced-news.

Chen, Wei-Fan, Benno Stein, Khalid Al-Khatib, and Henning Wachsmuth. "Detecting Media Bias in News Articles Using Gaussian Bias ... - Arxiv." Detecting Media Bias in News Articles using Gaussian Bias Distributions. Accessed April 20, 2023. https://arxiv.org/pdf/2010.10649.pdf.

"Gensim: Topic Modelling for Humans." Radim Å˜ehÅ¯Å™ek: Machine learning consulting, December 21, 2022. https://radimrehurek.com/gensim/.

"Let's Build from Here." GitHub. Accessed April 20, 2023. https://github.com/.

"Media Bias/Fact Check News." Media Bias/Fact Check, July 22, 2021. https://mediabiasfactcheck.com/.

"Pandas." pandas. Accessed April 20, 2023. https://pandas.pydata.org/.

Pritchard, Jonathan K, Matthew Stephens, and Peter Donnelly. 2000. "Inference of Population Structure Using Multilocus Genotype Data." Genetics 155 (2): 945–59. https://doi.org/10.1093/genetics/155.2.945.

Samantha D'Alonzo and Max Tegmark, "Machine-Learning Media Bias," arXiv.org (Dept. of Physics and Institute for AI & Fundamental Interactions, August 31, 2021), https://arxiv.org/abs/2109.00024.

"Spacy · Industrial-Strength Natural Language Processing in Python." · Industrial-strength Natural Language Processing in Python. Accessed April 20, 2023. https://spacy.io/.

"Simplified Text Processing." TextBlob. Accessed April 20, 2023. https://textblob.readthedocs.io/en/dev/.

Zhang, Xinliang. 2022. "Launchnlp/BASIL." GitHub. July 17, 2022. https://github.com/launchnlp/BASIL.

XIII. Appendices