An Analysis of Political Bias in News Media Through the Use of Natural Language Processing

Manuel Nunez

University of Florida

Summa Cum Laude Honors Thesis

Supervisory Board:

Bonnie Dorr, Zoey Liu, Sonja Schmer-Galunder

Table of Contents

A. Abstract

This thesis examines the categorization of political sentiment within US news media, contrasting the performance of rule-based and deep learning Natural Language Processing (NLP) models. A rule-based sentiment analysis model was developed to quantify the polarity and intensity of political bias, targeting sentiment towards nouns by analyzing associated adjectives and verbs. This model, grounded in coreference resolution and "closest POS" algorithms, creates a vectorized sentiment representation for each political orientation. Conversely, a deep learning approach utilizing a Convolutional Neural Network (CNN) was implemented to assess raw text through various neural layers that employ dimensional reduction without any prior sentiment extraction. The thesis advocates for a hybrid model, leveraging the strengths of both techniques for multidimensional data analysis. The research culminates in an application of these models, tracking sentiment progression in major news outlets over the 21st century. Finally, the work explores the practical utility of both models by graphing the sentiment biases of CNN and FOX News. This empirical examination further distinguishes between the correctness of each model. By analyzing vast quantities of news content and offering a nuanced understanding of political sentiment, this thesis provides significant insights into the composition of political bias in news media and contributes to the ongoing discourse on computational methods in media analysis.

B. Introduction

I. Project Overview

This paper explores the use of different Natural Language Processing Models in the categorization of political sentiment using articles from several US news outlets. The key focus of the project is dedicated to the implementation of a rule-based sentiment analysis model that aims to distinguish and quantify the magnitude and direction of political bias through techniques that isolate the sentiment expressed by a collection of articles. This model is developed in reference to a benchmark defined by a deep learning model, which assumes a more flexible, but opaque, approach to classification. The paper aims to comment on the weaknesses and strengths of both models and argues for a hybrid approach in the categorization of multi-dimensional data. It concludes with an application of the models, where both are used to track the progression of political sentiment in major news outlets throughout the 21st century.

The sentiment analysis model focuses on isolating the sentiment expressed towards nouns through the adjectives and verbs that describe them. It considers both improper and proper nouns (named entities) when evaluating sentiment expression. To attain a clear and comprehensive interpretation of expressed sentiment, the model employs coreference resolution to enrich the text by substituting references to named entities with the named entities themselves. The task of POS referencing, which in this case consists of locating the nouns referenced by verbs and adjectives, is accomplished through a simple “closest POS” algorithm which is shown to have an acceptable precision metric.

Ultimately, a vectorized representation of the sentiment expressed towards all nouns present in articles corresponding to one political leaning is composed, where each dimension is defined by a unique noun. Sentiment is quantified using the valence scores of all verbs and nouns considered. The classification process then consists of composing a vector for a batch of articles whose bias we want to

determine and taking the cosine distance between this vector and the vector created for each political leaning. Probabilities for each of the three political leanings are then provided as output in proportion to each cosine distance found.

The deep learning model employed is meant to counter the sentiment analysis model, by considering raw text as input as opposed to dissecting and extracting meaning from it first. A Convolutional Neural Network is used which takes in as input a tokenized and lemmatized article and applies embedding, dense, and flattening layers to reduce the dimensionality of the data and find meaningful recurrent patterns. The model's resulting performance metrics, considering its opaque architecture, pose a series of questions regarding **how** it is classifying inputs and whether its considerations in classifying them fall past the scope of political sentiment.

Lastly, we explore an application of both models by exploring the political bias expressed by the country's two largest and most controversial news outlets, CNN and FOX News. We graph the bias provided as output for both outlets over the past four years using large batches of articles for each three-month period. This exploration aims to further distinguish between the practical use of both models implemented.

II. Past Work and Research

Past efforts in the recognition of media bias define bias as a situation in which one supports or opposes someone or something in an unfair way because they are influenced by personal opinions or an unfair preference for one viewpoint. Notably, bias is not merely an expression of opinion, but instead the distortion or partial presentation of information to convey and support a belief. Pertinent to this exploration is the expression of bias through ideology, which "reflects a news outlet's desire to affect reader opinions in a particular direction" (Mullainathan & Shleifer, 2002) This desire is deliberate, and its

instances often underscore a general purpose where the ideology presented finds its roots. This leads to "the same bias being frequent and consistent for each media outlet".

Explorations of media bias differentiate between three main categories: coverage bias, gatekeeping bias, and statement bias (Saez-Trumper, Castillo, & Lalmas, 2013). Coverage bias is noted through the prevalence of different topics covered by an outlet. While the presentation of coverage may not be biased itself, the partiality of the information accessible through an outlet can lead readers to have a distorted perception of events. Gatekeeping bias expands on coverage bias and delves into the intentional rejection or selection of stories related to relevant entities. Reporting the successes of a political candidate while refusing to report their failures is a common form of gatekeeping bias. Lastly, statement bias refers to the use of rhetoric, commonly through "labeling and word choice" (Hamborg, Zhukova, & Gipp, 2019), to describe entities. The rule-based sentiment recognition model most directly recognizes this form of bias through the identification of words used to describe entities in text.

Ultimately, the presence and detection of bias is heavily circumstantial and subjective. There is truly "no agreed-upon definition of what constitutes bias in journalism" (Francisco-Javier Rodrigo-Ginés, Jorge Carrillo-de-Albornoz, Laura Plaza, 2023) This exploration assigns associated bias with the political leaning of different new outlets, as classified by credible academic sources. However, this ground truth is founded on generalizations that fail to capture the nuances and particular instances of expressed bias.

Computational approaches to bias detection in media have historically taken the form of binary or multi-class classification problems. The exploration presented conforms to the latter as it aims to map an input to one of three political leanings. Non-deep learning models, such as the sentiment recognition model implemented, have paired "SVM, Logistic Regression, and Random Forest" techniques with the extraction of hand-crafted features to detect bias in text. "The performance of such a method is heavily influenced by feature selection" (Cruz, Rocha, & Cardoso, 2019) which encompasses lexical and

semantic characteristics of text. Approaches to bias detection using such methods lost popularity post-2010 due to the advent of deep learning.

The work of Hube and Fetahu (2018) in *Detecting Biased Sentiment in Wikipedia* assumes a rule-based approach to the detection of bias through a supervised learning model that "relies on an automatically created lexicon of bias words, and other syntactical and semantic characteristics of biased statements." The model built is meant to classify sentence-length text. The paper defines a biased article as one flagged with NPOV (Neutral Point of View) by site editors. Such a flag indicates the presence of biased text. While the classification of input is binary, the features of bias explored are comparable to those considered by the sentiment recognition model.

The work focuses on first composing a lexicon of politically charged words. These are found in Conservapedia, which is a "Wiki shaped according to right-conservative ideas." Word2Vec is used to augment the lexicon by adding the words closest to all biased words initially found in Conservapedia. Although biased words are frequently found in biased sentences, a high degree of ambiguity can stem from the context in which a biased word is detected. To avoid this shortcoming fourteen additional features are considered. Notably, POS trigrams, sentiment values, and LIWC features are incorporated. A Random Forest architecture is employed to classify inputs and an accuracy of 74% is found in the classification of politically biased sentences.

The self-learning mechanisms of deep learning models allow for the automatic extraction of features and attention to the sequential nature of text. Past research in bias detection using deep learning has mainly employed Recurrent Neural Networks (RNNs) and Transformer methods. Transformers are considered to outperform RNNs in text classification as they "do not have an internal state, instead, Transformers use the so-called self-attention mechanism to model the sequential structure of a sentence" (Vaswani et al., 2017). Comparable to the limits of hand-crafted feature extraction in rule-based models, deep learning models are limited by their reliance "on the distribution

of low-level lexical information", which is often not implicative of political bias. The Convolutional Neural Network Model (CNN) later implemented is thought to suffer from this issue.

The work of Chen, Al-Khatib, Stein, and Wachsmuth (2020) in *Detecting Media Bias in News Articles using Gaussian Bias Distributions* attempts to circumvent the constraints of hand-crafted feature extraction while also avoiding the misdirected attention of deep-learning networks. The paper considers second-order information, which translates to "probability distributions of the frequency, positions, and sequential order of lexical and informational sentence-level bias in a Gaussian Mixture Model". In simpler terms, the approach aims to process text and generate inputs such that features possibly utilized by deep learning networks will be indicative of political bias expression. This thesis argues for an approach of the same nature.

We implement models that lie at the extremes of rule-based and deep-learning architectures. The rule-based model limits itself to quantifying one major feature, namely sentiment expressed towards entities, and does not employ machine learning. The deep learning CNN model is not focused on sequential patterns, as are RNNs and Transformers, and is instead given full freedom to classify article inputs. By assessing how both models perform against training data sets and external articles from separate news outlets, the shortcomings of each will be noted and a case will be made for architectures that combine the features of both.

C. Software Design, Development, and Implementation

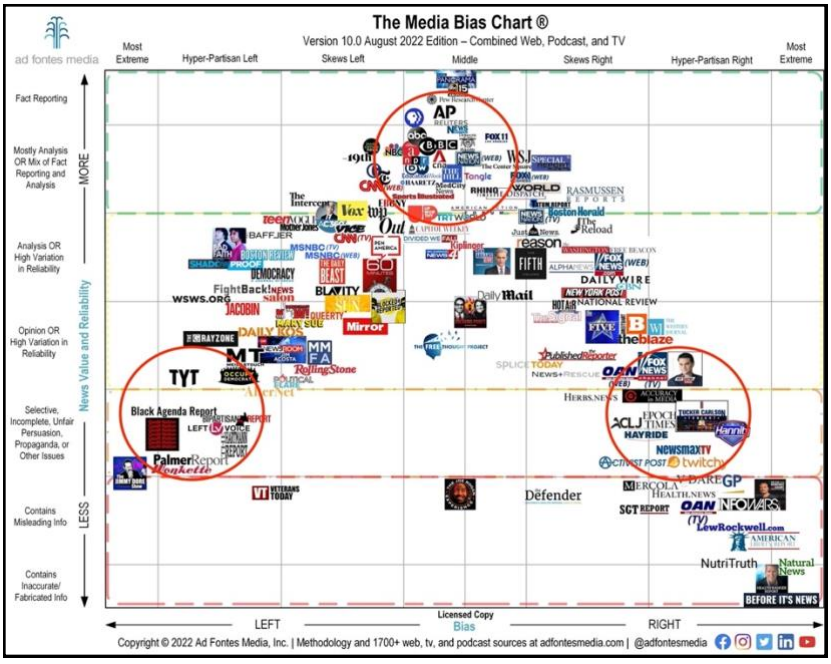
I. Data Sourcing

The news feed used to implement the models discussed throughout this exploration was obtained through The Newscatcher API. The Newscatcher API provides flexible methods to query data through a variety of parameters that specify attributes such as news sources, keywords, topics, etc.

Both models are premised on the idea that by exploring outlets with political biases that are either very extreme or objectively centered, three distinct categories of bias can be considered such that a ground truth can be established. A political leaning of far-right, center, or far-left can then be assigned to each of the three groups of articles queried.

To begin sourcing the data, the available news sources provided by the API are queried. Political bias charts are then researched, and trustworthy sources are found through which to choose an eclectic group of news outlets. Namely, the media bias chart shown below (University of Central Oklahoma, 2022), which classifies media bias in reference to political bias and reliability, is used. Focusing solely on the political bias dimension, the outlets chosen are all located within the red circles shown in the image. **PBS, AP News, and News Nation Now** are chosen as center outlets, **Palmer Report and Bipartisan News** are chosen as far-left outlets, and **VDare, News Max, and Ricochet** are chosen as far-right outlets.

Figure 1: News Outlet Spectrum



The program logic querying data aims to minimize query calls while concurrently sourcing a comprehensive amount of data for the time interval explored. This task is further complicated by query calls that return articles with no text to examine. The query loop shown below is used together with the function “getArticles” that performs the API call. The loop deals with each news outlet separately and considers ten separate three-month intervals from which to source articles. It progressively builds a data frame with articles found and only adds those containing text. To reserve API calls, it stops considering a given time interval after more than three thousand articles have been found. Lastly, the loop saves each outlet’s data frame to a CSV file.

Although it would be optimal to consider a greater number of sources for each political category, access to outlets is limited by the number of API calls available and the outlets to which the API has access. Furthermore, to ensure that the news feed queried was evenly distributed, the articles examined are limited to a three-year range from January 1st,2021 to December 31st,2023.

Figure 2: News Query Loop Code

```
# Define all news sources being queried
sources = ['newsmax.com','vdare.com', 'newsnationnow.com', 'palmerreport.com', 'bipartisanreport.com', 'pbs.org', 'newsnationnow.com', 'apnews.com']
# Define all time intervals foor querying
dates = [("2021/04/01", "2021/06/30"), ("2021/07/01", "2021/09/30"),
        ("2021/10/01", "2021/12/30"), ("2022/01/01", "2022/03/30"),
        ("2022/04/01", "2022/06/30"), ("2022/07/01", "2022/09/30"),
        ("2022/10/01", "2022/12/30"), ("2023/01/01", "2023/03/30"),
        ("2023/04/01", "2023/06/30"), ("2023/07/01", "2023/09/30")]
# for each news outlet
for source in sources:
    df = pd.DataFrame()
    # For each interval
    for interval in dates:
        # Set start and end dae and pull articles from every third query page
        start_date = interval[0]
        end_date = interval[1]
        pages = getArticles(source,start_date, end_date,"1")["total_pages"]
        step = pages // 3
        for page in range(1, pages, step):
            articles = getArticles(source,start_date, end_date,str(page))["articles"]
            temp_df = pd.DataFrame(articles)
            temp_df = temp_df[temp_df['summary'] != '']
            df = df.append(temp_df, ignore_index=True)
            # set cap on  articles per time frame
            if len(df) > 3000:
                    break
    file_name = source[:len(source)-3] + 'csv'
    clean_df = df[['title', 'published_date', 'clean_url', 'summary']]
    clean_df.to_csv(file_name)
```

Figure 3: News Query Code

```
def getArticles(source, start_date, end_date, page):
    search_url = "https://api.newscatcherapi.com/v2/search"
    search_querystring = {"q":"*","lang":"en","countries":"US", "topic":"politics",
                          "sources":[source],"page":page, "from":start_date, "to":end_date}
    headers = {
        "x-api-key": api_key
        }
    response = requests.request("GET", search_url, headers=headers, params=search_querystring)
    articles_dic = json.loads(response.text)
    return articles_dic
```

II. Exploratory Data Analysis

While the query loop is meant to source equal amounts of data for each outlet, the process results in an unequal pull of data for each news outlet. The images below showcase the distribution by outlet for each three-month period considered. It's clear that some outlets have significantly more articles queried than others and that the number of articles for each three-month interval considered is similarly unevenly distributed.

Figure 4: Left-Leaning Outlets Distribution

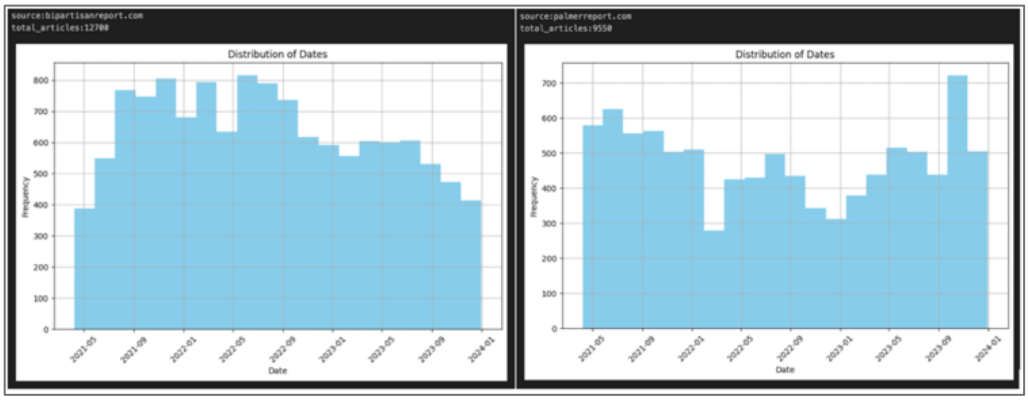


Figure 5: Right-Leaning Outlets Distribution

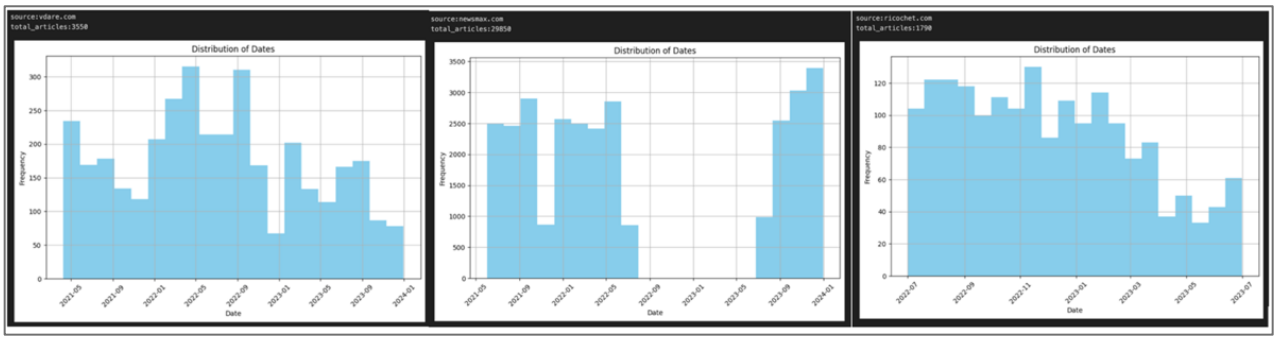
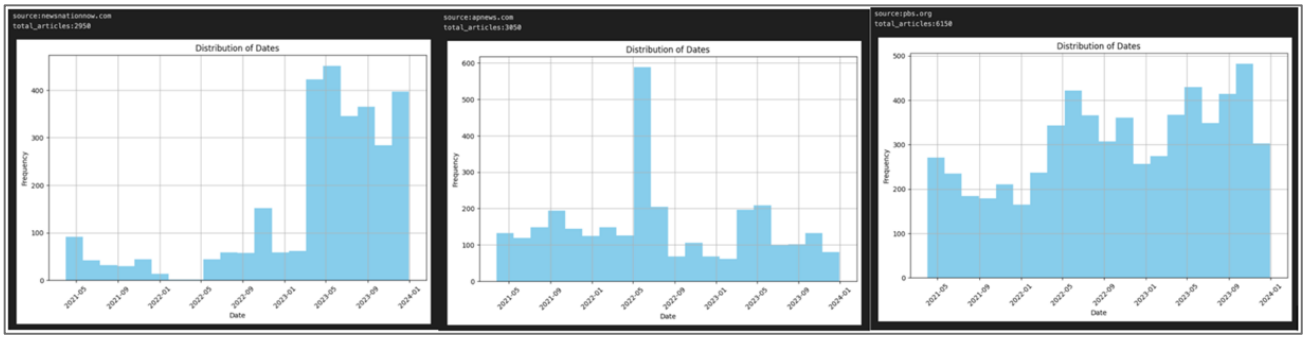


Figure 6: Center-Leaning Outlets Distribution



The uneven distribution of articles and outlets must be resolved to avoid future model bias. To do so, each category is truncated to only contain ten thousand articles. Additionally, because all API calls have been exhausted, this work strives for an even distribution for the aggregate of all articles pertinent to each class, instead of an even distribution for each outlet. After methodically deleting article entries within certain time intervals for different outlets, the resulting distributions for each category are shown below. While the distributions are far from perfect, they are substantially improved.

Figure 7: Center Outlets Aggregate Distribution

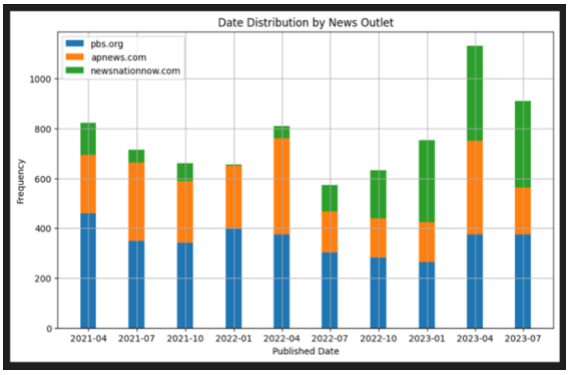


Figure 8: Right Outlets Aggregate Distribution

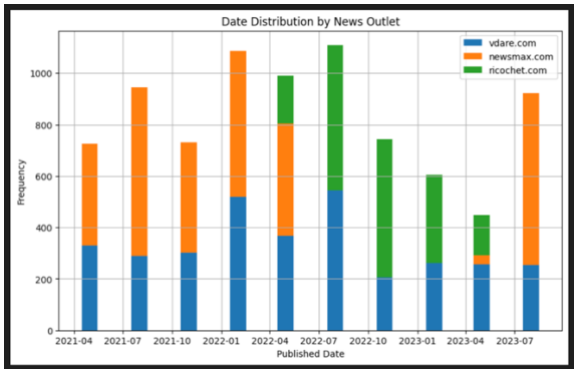
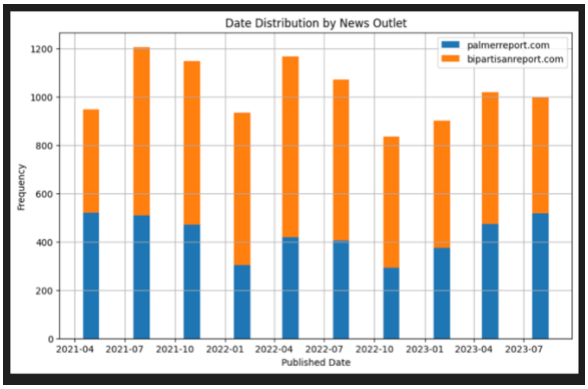


Figure 9: Left Outlets Aggregate Distribution



III. Data Pre-Processing

Data preprocessing is largely specific to each model implemented, but some general corrections are necessary for each model to avoid classification based on textual characteristics specific to each outlet. For example, most Palmer Report articles conclude with a string that reads as follows “Facebook Share/nTwitter Share/nEmail this article/nPrint Article”. Initially, this string and others of the same sort are removed from all articles as they create irrelevant patterns that both models could leverage to classify articles. However, the process of removing such strings cannot be generalized and consists of a tedious case-specific analysis. Accordingly, given that most strings that need removal are within the beginning or end of the articles, it is chosen to instead disregard the first three and last three sentences

of each article. While this practice risked ignoring worthwhile portions of the article, it was successful in removing a great majority of irrelevant strings.

IV. Sentiment Analysis Model: Implementation

The implementation of the rule-based sentiment analysis model aims to determine political bias by extracting and quantifying the sentiment expressed towards nouns through the verbs and adjectives that reference them. This is done through coreference resolution, dependency parsing, Part-of-Speech (POS) reference resolution, sentiment vectorization, and cosine distance as the ultimate classification metric. Each step is further explained below.

1. Coreference Resolution

Coreference resolution is used to enrich the text by substituting all references to named entities with the named entity itself. This avoids grouping sentiment for lexically equivalent nouns that refer to different entities. Two common examples seen throughout the data set are the referencing of named entities through pronouns and the referencing of named entities through improper nouns. For example, the text “John is gifted. He was always good at math.” is changed to “John is gifted. John was always good at math”. We can now associate John with the adjectives “good” and “gifted” as opposed to associating “gifted” with “he”. This stops the pronoun “he” from being considered. Otherwise, the resulting sentiment dictionary will point at an average of the sentiment expressed towards the different entities referenced by the pronoun. This is misleading and substantially weakens the model.

To perform coreference resolution we employ the Spacy coreference resolution model which “is an end-to-end neural system applicable across a wide variety of entity coreference problems.” The model abandons the “use of pre-trained parsers, mention detectors, and other learned or rule-based components during training and inference” and leverages large pre-trained transformer models to

instead “produce pairwise coreference scores between contiguous sequences of tokens indicating how likely is it that they belong to the same cluster.” The clustering is evaluated by comparing the grouping of mentions produced by the system to the ground truth clustering provided in the test data. Put in simpler terms, the model identifies entities in text and then determines the likelihood that they are being referenced by each improper noun, pronoun, and other referencing nouns around them.

The following function is used to take in a document and output a string with all references resolved. The function initially iterates through each cluster and creates a dictionary where the ID of each token referencing another is associated with the entity it is referencing. The sentence is then rebuilt substituting references for their referenced entity.

Figure 10: Coreference Resolution Code

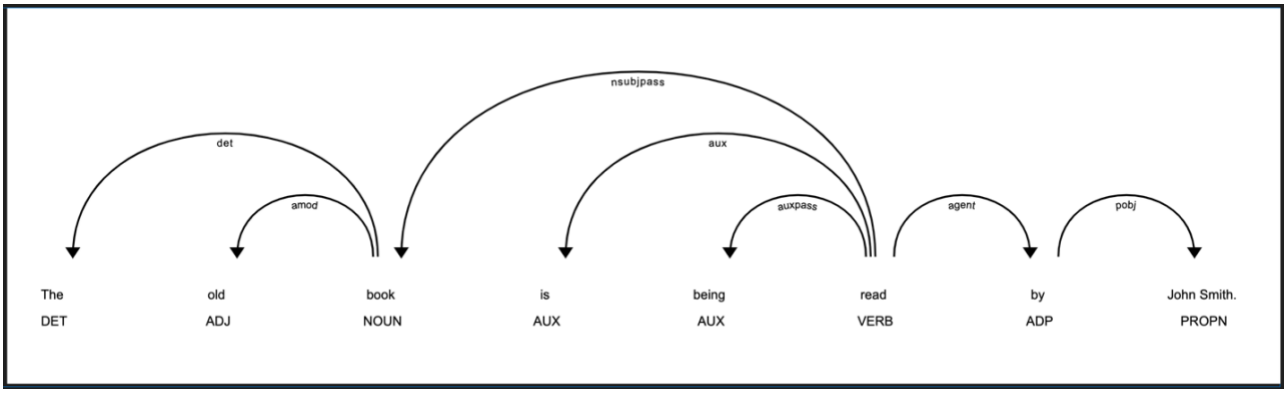
```
def resolve_references(sentence):
    doc = nlp(sentence)
    token_mention_mapper = {}
    output_string = ""
    clusters = [
        val for key, val in doc.spans.items() if key.startswith("coref_cluster")
    ]

    # Iterate through every found cluster
    for cluster in clusters:
        first_mention = cluster[0]
        # Iterate through every other span in the cluster
        for mention_span in list(cluster)[1:]:
            # Set first_mention as value for the first token in mention_span in the token_mention_mapper
            token_mention_mapper[mention_span[0].idx] = first_mention.text + mention_span[0].whitespace_

            for token in mention_span[1:]:
                # Set empty string for all the other tokens in mention_span
                token_mention_mapper[token.idx] = ""

    # Iterate through every token in the Doc
    for token in doc:
        # Check if token exists in token_mention_mapper
        if token.idx in token_mention_mapper:
            output_string += token_mention_mapper[token.idx]
        # Else add original token text
        else:
            output_string += token.text + token.whitespace_

    return output_string
```

2. Dependency Parsing and Part of Speech Reference Resolution

With coreferences resolved, the model now tasks itself with correctly associating verbs and adjectives with their corresponding noun. To do so we make use of Spacy dependency trees. A

dependency tree is a graphical representation of the syntactic structure of a sentence that shows how words in the sentence are related to one another. The tree consists of nodes and directed edges, where each node represents a word, and the edges represent the dependencies between the words. Once a sentence is turned into a Spacy dependency tree, one can traverse the tree starting at its head and move down to its children tokens. To incorporate the advantages of coreference resolution into the dependency trees, parsed sentences are retokenized to make multi-word named entities a single token. An example of the sentence "The old book is being read by John Smith." is shown below.

Figure 11: Dependency Tree: Retokenizing Named Entities



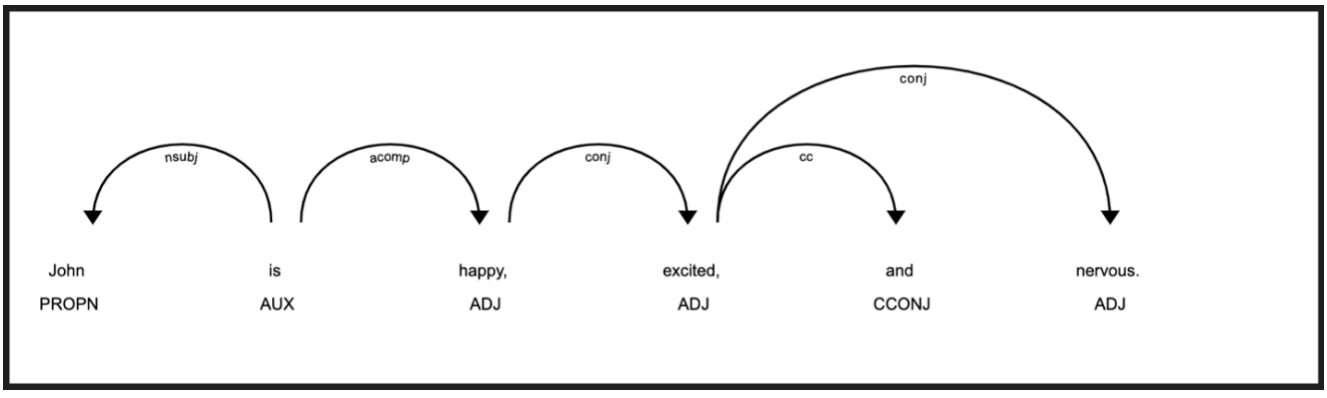
The complexity of dependency paths between related parts of speech makes it a complicated and expensive task to find all relations between nouns and their referencing adjectives and verbs with complete accuracy. To avoid a computationally expensive approach while still attaining a deep understanding of sentiment, this task focuses on an approach that balances the likelihood of identified relations being accurate and the identification of a high number of present relations.

An initial method parses a dependency tree and finds all AMOD (adjective modifier) dependencies, it then climbs up the tree by one node and if the token found is a noun the relation between this token and the adjective corresponding to the AMOD relationship is added. The

dependency tree above displays this relation between “book” and “old”. The algorithm implemented has a complexity of O(N) for a sentence of length N, given N tokens in a sentence are traversed linearly, and for every token with dependency AMOD, an O(1) operation checks for the presence of a noun above it. This method presents a high-precision performance metric, meaning it almost exclusively identifies correct relations. However, its scope is very limited given it cannot be applied to noun-to-verb relations and that several correct relations between nouns and adjectives are ignored.

To avoid a shallow interpretation of sentiment, more leeway is made for the identification of incorrect relations through an alternative approach. This method is founded on the idea that a sentence can contain a noun with no verb or adjective describing it, whereas the presence of a verb or adjective in a sentence almost always implies the presence of a noun. Accordingly, as opposed to finding relations stemming from nouns to verbs and adjectives, the method finds relations stemming from verbs and adjectives to nouns. The algorithm then associates each verb or adjective with whichever noun is closest to it, regardless of where it is located. The dependency tree shown below displays an example where the first method presented finds no relations whereas the current approach finds all relations present (in this case between nouns and adjectives).

Figure 12: Dependency Tree: One Noun, Multiple Adjectives



The algorithm uses bottom-up dynamic programming to reduce the method's complexity. It first carries out an O(N) parse of the dependency tree starting at the head and progressively updates an array storing pair objects of type (int, string) holding the distance between each node and the closest noun below it as well as the string of the noun found. A token's index in the array corresponds to its index in the sentence. If there is no noun below the current node the pairs retain their default value (-1, ""). Once the memoization array has been filled in, the tokens are traversed linearly with complexity O(N). In the case where relations are being found between nouns and adjectives, for each adjective encountered, we check how far away the closest noun below it is. This distance is labeled as D, which we obtain from the memoization array. We then traverse up recursively a number of times equal to D – 1 to look for closer nouns that could branch off or be contained in parent nodes. If the adjective being considered has a memoization value of (-1, ""), the algorithm will recursively check parent nodes until it encounters the root. Each of these checks is of complexity O(N) such that the entire algorithm is of complexity O(N) + O(N^2) which can be simplified to O(N^2). The functions defining the algorithm are shown below.

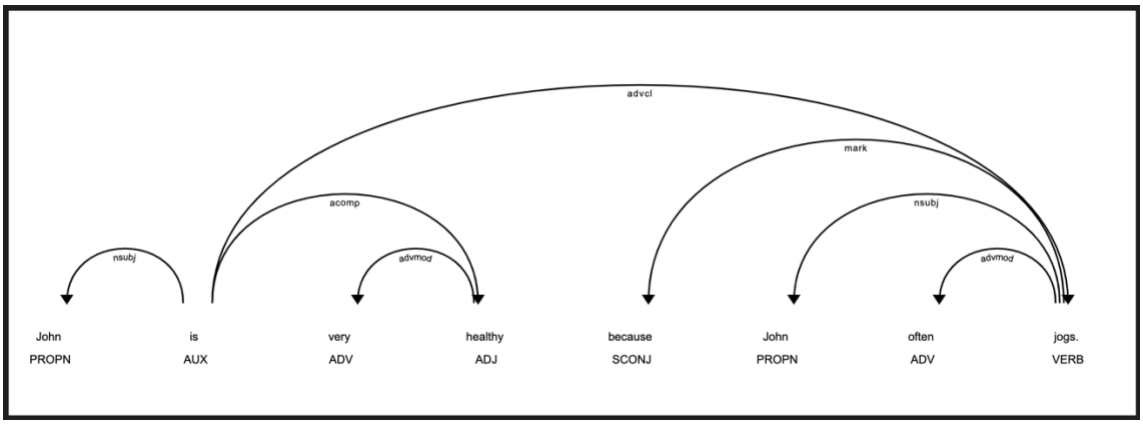
Figure 13: POS Reference Resolution Code

```
def find_distances(sentence, from_pos, to_pos):
    sent_dic = {}
    # Create memoization array
    mem = [(-1, "")] * len(sentence)

    #mark closest predecessors
    mark_down(sentence.root, to_pos, mem)

    #consider succesors and finalize dictionary
    for token in sentence:
        if token.pos_ in from_pos:
            # cretae dictionary entry for token
            if token.text not in sent_dic:
                sent_dic[token.text] = []

            if token != sentence.root:
                mark_up(token, mem, token.head, 1)

            sent_dic[token.text].append(mem[token.i][1])

    return sent_dic
```

```
# check parent node recursivley as long as current distance from a noun isnt surpassed
def mark_up(ref, mem, token, dist):
    closest = mem[token.i][0]
    if closest != -1 and (closest + dist < mem[ref.i][0] or mem[ref.i][0] == -1):
        mem[ref.i] = (mem[token.i][0] + dist, mem[token.i][1])

    if token.dep_ != "ROOT":
        mark_up(ref, mem, token.head, dist + 1)


# mark all entires in memoization array with closest noun succesor
def mark_down(token, to_pos, mem):
    if token.pos_ in to_pos:
        mem[token.i] = (0, token.text)

    dist = (10000, "")
    for child in token.children:
        mark_down(child, to_pos, mem)
        closest = mem[child.i][0]
        if closest != -1 and closest < dist[0]:
            dist = (closest, mem[child.i][1])

    if dist[0] != 10000 and token.pos_ not in to_pos:
        update = (dist[0] + 1, dist[1])
        mem[token.i] = update
```

For further clarity, consider the sentence “John is very healthy because he often jogs" to which coreference resolution will be applied. The resulting sentence is “John is very healthy because John often jogs”. We can see the dependency tree for the enriched sentence below. A memoization array of length eight is initialized and the “mark_down” function fills in the values as shown. Notice, for example, how the second entry contains a distance of 1 since the noun “John” is a child of the auxiliary “is”.

Figure 14: Dependency Tree: Algorithm Walkthrough



Memoization: [(0, “John”), (1, “John”), (-1, “”), (-1, “”), (-1, “”), (0, “John”), (-1, “”), (1, “John)]

Starting with verbs, the algorithm then considers index 7 which corresponds to the verb “jogs” and entry (“1”, John). D then equals one and the algorithm looks up D – 1 times to find a closer distance. However, since D equals 1, no parents are considered, and we find that “jogs” corresponds to “John”. A similar procedure is followed for adjectives. For example, index three in the array corresponds to the adjective “healthy” and entry (-1, “”). Given that there is no noun successor to “healthy”, we will instead recursively traverse parent nodes until the root and retain the shortest distance. We will first consider the auxiliary “is” and find it corresponds to entry (1, “John”). Since we are one edge above “healthy” and “is” is one edge above “John”, we know that “John” is two edges away from “healthy”. Recursively checking the parent of “is” is, in this case, unnecessary since any shorter distance would have to be less

than two edges, and the parent of “is” is two edges away from “healthy”. The algorithm then determines that “healthy” corresponds to “John”.

This method is not all-encompassing and at times ignores correct relations and misidentifies incorrect ones. One of the primary shortcomings of the algorithm is that it assumes a one-to-many relationship between nouns and their referencing part of speech (either verbs or adjectives). This relationship can at times be many-to-many or many-to-one. For example, the sentence “John is happy and excited” has a one-to-many relationship between its noun (John) and its adjectives (happy and excited). The algorithm will correctly identify both relationships. However, the sentence “ John and Peter are happy”, which has a many-to-one relationship between its nouns (John and Peter) and its adjective (happy), will cause the algorithm to only identify one relationship between the adjective “happy” and its closest noun “Peter”.

To verify the algorithm’s viability for POS reference resolution a data set of one hundred random sentences is employed and all relations found by the algorithm are compared to the correct human annotated relations. The tests reported a precision score of 0.82 for relations between adjectives and nouns and a precision score of 0.75 for relations between verbs and nouns, meaning that a high percentage of the relations found are correct relations. Recall scores of 0.78 and 0.68 were found. This indicates that the algorithm can successfully detect a majority of all relations present. The balance found between recall and precision suits the model’s demand for a high and confident degree of information extraction regarding POS references.

3. Sentiment Vectorization and Cosine Distance Computation

Sentiment vectorization consists of first defining an N-dimensional space through a vector of length N where each dimension pertains to a unique noun found in the corpus of articles. Each group of news outlets explored is mapped onto this N-dimensional space. This allows for an article to be mapped

onto this space as well, by extracting the sentiment expressed towards all the nouns present within it and considering only those nouns that are found in the original training corpus (which defined the original vector space). Once an article has been vectorized, the cosine distance between this vector and each vector defining the three political leanings is computed. The vector closest to the article vector will correspond to the political leaning that the article primarily expresses.

Using dependency parsing and the POS Reference Resolution algorithm to identify adjective and verb references to nouns in text, a dictionary is assembled associating each noun in an article with a list of the nouns and verbs referencing it. The valence score of each verb and adjective is extracted using the Python TextBlob library. Valence, in the context of language, defines the degree of positive or negative sentiment expressed. Hence, we define the sentiment expressed towards a noun in an article by taking the average valence score of all nouns and verbs referencing it. By doing this for each article within a political leaning and merging the dictionaries for all articles, we obtain one final dictionary whose keys consist of all the unique nouns in the corpus and point at the average valence score expressed towards all mentions of each noun.

Ultimately, following this process for each of the three groups of articles results in three dictionaries with each holding the nouns found in their respective corpus. To compare groups, these dictionaries are consolidated to hold references to the same group of nouns. To do so, a separate dictionary is composed assigning an integer identifier to each noun found throughout all three training corpora. If N distinct nouns are found, a vector of length N is defined for each of the three groups of articles. For each group, the sentiment expressed towards each noun is assigned to index K where K is the identifier of that noun. For indexes corresponding to nouns not present in a group's corpus, a sentiment score of 0 will be assigned.

Ultimately, there are three vectors of equal length that map onto an N-dimensional space. Each vector is interpreted as the sentiment expressed towards all nouns present in the entirety of the corpus.

Lastly, to classify an article the valence scores for all nouns are computed. These scores are added onto a vector of length N, disregarding nouns that were not in the original training corpus. The cosine distance between this vector and each of the three vectors originally computed is taken and the political leaning that the closest vector identifies with is assigned to the article being classified.

Consider the following simplified example where three dictionaries are composed based on the corpus of each bias group:

Left Bias Dictionary: {Trump: -0.70, IRA: 0.50, Israel: 0.10, Immigrant: 0.30}

Right Bias Dictionary: {Trump: 0.80, IRA: -0.10, Israel: 0.80, Vaccine: -0.50}

Moderate Bias Dictionary: {Trump: -0.20, IRA: 0.10, Israel: 0.30, China: -0.1}

An additional dictionary is then implemented to assign an identifier to each noun encountered:

Identifier Mapping: { Trump: 0, IRA: 1, Israel: 2, Immigrant: 3, Vaccine: 4, China: 5}

This dictionary is then used to translate each of the three initial dictionaries into vectors of length 6. Notice how nouns absent in a group's corpus are assigned a score of 0:

Left = [-0.70, 0.50, 0.10, 0.30, 0.0, 0.0]

Right = [0.80, -0.10, 0.80, 0.0, -0.50, 0.0]

Moderate = [-0.20, 0.10, 0.30, 0.0, 0.0, -0.1]

A dictionary representing an article to be classified is shown below as well as its representation as a vector of length 6. Notice how the noun "Canada", which isn't present in the training corpus, is not shown in the vector used for classification:

Article Dictionary: {Trump: -0.30, Immigrant: 0.10, Canada: 0.05}

Article vector = [-0.30, 0.0, 0.0, 0.10, 0.0, 0.0]

The article can then be classified by taking the cosine distance between its vector and each of the three vectors pertaining to each political leaning:

cosine distance (article, left): 0.17

cosine distance (article, right): 1.61

cosine distance (article, Moderate): 0.51

As expected, given the negative sentiment expressed towards Donald Trump, a major representative of the republican party, the article's vector is closest to the left bias vector. We thus classify the article as having a primarily left-leaning bias.

V. Convolutional Neural Network Model: Implementation

A Convolutional Neural network was chosen as opposed to another type of neural network because its architecture allows for the interpretation of local dependencies, can handle variable-length text, and can attain high levels of accuracy without demanding feature extraction on behalf of the programmer. To approach text classification in a manner that distinctly contrasts with the methods employed by the rule-based sentiment analysis model, the neural network built was given complete freedom in its methodology used for classification. Its implementation aims to capture the concept of black-box functionality. The model initially requires standard text preprocessing and tokenization. Once text has been cleaned and tokenized, the model's architecture is specified, and a training loop gradually corrects its accuracy. The model was largely inspired by an Atmosera article (Prosise, 2023) which specifies the architecture used. Each step is further explained below.

1. Preprocessing and Tokenization

The implementation of the model begins by combining all three data sets corresponding to the three political leanings explored. As specified beforehand, the articles in the data frames are stripped of their first three and last three sentences to avoid irrelevant strings. Each article then has its stop words removed. Stop words are words in a sentence that only serve a functional purpose and don't add meaning to expression (i.e. words like "then" and "if"). Once all stop words are removed, a tokenizer

from the Keras library was used to assign each unique word an integer identifier. This is necessary for a neural network to interpret model input and find patterns in the data from which to draw classifications.

2. Architecture and Training

The model begins with an Embedding layer, which transforms integer-encoded vocabulary into dense vectors of fixed size This layer is efficiently deals with the vast vocabulary of text data. It provides a meaningful representation of words which capture semantic similarities based on their context within the corpus. The dense vector representation allows the model to interpret text input in a format conducive to identifying patterns relevant to classification tasks.

Following the Embedding layer are two sets of one-dimensional convolution layers and Max Pooling layers. The convolution layers apply convolutional operations to the embedded word vectors, using filters to extract local patterns (such as the presence of specific n-grams) that are indicative of the text's class. The rectified linear unit (ReLU) activation function ensures that the model captures nonlinear relationships between these features. Each convolution layer is followed by a Max Pooling layer, which reduces the dimensionality of the data by retaining only the most prominent features, thus improving computational efficiency, and helping to prevent overfitting.

After the second convolution and pooling sequence, a Global Max Pooling layer aggregates the most significant features from across the entire text, ensuring that the model's final predictions are informed by the most impactful elements of the input data. The architecture culminates in a Dense layer with a Soft Max activation function, which maps the extracted features to probabilities across the three classes, allowing the model to quantify and output the distinctions noted between classes explored. The model is trained through five separate epochs which make use of a validation dataset to progressively increase its accuracy. The code shown below defines the model’s architecture:

Figure 15: CNN Model Architecture

```
num_classes = 3

model = Sequential()
model.add(Embedding(100000, 32))  # Embedding layer
model.add(Conv1D(32, 7, activation='relu'))  # First convolutional layer
model.add(MaxPooling1D(5))  # First pooling layer
model.add(Conv1D(32, 7, activation='relu'))  # Second convolutional layer
model.add(GlobalMaxPooling1D())  # Global max pooling layer
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.summary()
```

D. Results

Below we evaluate the performance of both models built throughout the exploration. We consider the precision, recall, and F1 scores for each of the three classes categorized. Precision measures the accuracy of positive predictions and Recall assesses the model's ability to identify all actual positive instances. F1-score is the harmonic mean of precision and recall, providing a single score that balances both concerns. A high F1-Score indicates that the model has a good balance between precision and recall. It should be noted that the results shown below define a model's performance in the classification of an article, not in the recognition of political bias. This distinction is further commented on when both models are applied to external news outlets.

I. Sentiment Analysis Model: Evaluation

The evaluation of the rule-based model was done using a portion of articles from each political leaning set aside for testing. Each article is applied coreference resolution and the sentiment expressed towards all nouns is quantified through a vector. This vector is then compared to each of the vectors defining the three political leanings explored. The article is assigned the political leaning of the vector found closest to the test vector.

Overall, the model was found to have an accuracy of roughly 53.6% . The following precision, recall, and F1-scores present the model's performance in reference to each classification group:

Figure 16: Rule-based Model: Metrics

	Precision	Recall	F1-Score
Left	0.44	0.75	0.55
Center	0.63	0.43	0.51
Right	0.54	0.42	0.47

The “Center” class has a precision of 0.63, which means that 63% of articles predicted as having left political bias are correctly classified, showing an acceptable level of accuracy. The “Left” and “Right” classes have lower precision scores of 0.44 and 0.54, respectively, which are lower than expected, but still signify that the correct class is chosen more often than the two other classes considered for both classes.

The “Left” class stands out with a recall of 0.75, indicating that 75% of the actual “Left”' instances are captured by the model's predictions. The “Center” class has a recall of 0.43, which is significantly lower, suggesting that the model isn’t great at detecting “Center” instances. The “Right” class has a similar recall of 0.42.

The “Left” class has the highest F1-Score of 0.55, which is not outstanding but falls past the 0.50 mark. The “Center” and “Right” classes have slightly lower F1-Scores of 0.51 and 0.47, respectively. These findings are summarized by the confusion matrix below:

Figure 17: Rule-based Model: Confusion Matrix

Confusion Matrix: Rule-Based Model			
Left	**932**	**617**	**569**
Center	**131**	**752**	**306**
Right	**168**	**370**	**645**
	Left	**Center**	**Right**

II. CNN Model: Evaluation

The convolutional Neural Network model is trained through five epochs and quickly attained an accuracy of 90%. No more epochs are used to avoid overfitting the model to its training data. The

plot below shows the progression of the model's training accuracy throughout the five training epochs.

Figure 18: CNN Model: Training Process

The model is then tested on the testing data set which encompasses 20% of the original data. Overall, the model is found to have an accuracy of roughly 89.7% . The following precision, recall, and F1-scores present the model's performance in reference to each classification group:

Figure 19: CNN Model: Metrics

	Precision	Recall	F1-Score
Left	0.95	0.96	0.96
Center	0.83	0.92	0.87
Right	0.89	0.76	0.82

The “Left” class has a precision of 0.95, which means that 95% of articles predicted as having left political bias are correctly classified, showing a high level of accuracy in these predictions. The “Center” and “Right” classes have lower precision scores of 0.83 and 0.89, respectively, which are still quite high but indicate a higher rate of false positives for these classes compared to the “Left” class.

The “Left” class again stands out with a recall of 0.96, indicating that 96% of the actual “Left”' instances were captured by the model's predictions. The “Center” class has a recall of 0.92, also high, suggesting that the model is quite good at detecting the “Center” instances. The “Right” class, however, has a lower recall of 0.76, meaning it missed more actual “Right” instances than the other classes.

The “Left” class has the highest F1-Score of 0.96, which is an outstanding result, indicating a very well-performing model for this class. The “Center” and “Right” classes have slightly lower F1-Scores of 0.87 and 0.82, respectively. These findings are summarized by the confusion matrix below:

Figure 20: CNN Model: Confusion Matrix

Confusion Matrix: CNN Model			
Left	**2035**	**48**	**36**
Center	**36**	**1317**	**74**
Right	**60**	**230**	**894**
	Left	**Center**	**Right**

III. Application and Insights

The applicability of both models is explored by classifying articles of news outlets not included in the training corpus. This is done to distinguish between a model’s ability to truly recognize bias, as opposed to differentiating between the outlets that trained them. The rule-based approach presents an attempt to target and extract features of text that express sentiment, such that the use of similar rhetoric or other features of an apolitical character are disregarded. The CNN model

assumes an opposite approach and is instead given complete freedom to differentiate between corpora through any means available.

As expected, the consideration of more lexical features results in the CNN more accurately differentiating between corpora. However, accurately distinguishing between groups of outlets does not translate to an accurate interpretation of bias. Conversely, regarding the rule-based model, a low accuracy score in the classification of articles is not indicative of an inability to classify bias. To solely judge the recognition of political bias, we abandon the outlets used to compose the training corpus, such that similarities in prose, structure, and other lexical features present within each grouping of outlets can no longer be leveraged for classification.

CNN and FOX News are the two outlets used to explore the applicability of the models. These outlets are chosen because they are two of the country's largest news media corporations and are widely acknowledged to land on opposing sides of the political spectrum. The opinions they express are expected to align closer with those conveyed by the right-leaning and left-leaning classes explored. However, the two outlets are comparable to the center-leaning outlets in that they are large and widely read news sources. This means that articles written by the outlets are likely to be closer to those written by center-leaning articles in style and structure. Articles from CNN and FOX are considered for the three-year period that the training corpus encompasses. Additionally, to add a layer of temporal analysis, batches for each three-month period within the three years explored are classified separately. This will allow for the evaluation of the model's predictions in reference to periods that correspond to different degrees of political tension in the country. The distribution of CNN and Fox articles used are shown below. Roughly 1500 articles are classified for each outlet.

Figure 21: CNN Articles: Distribution by Period

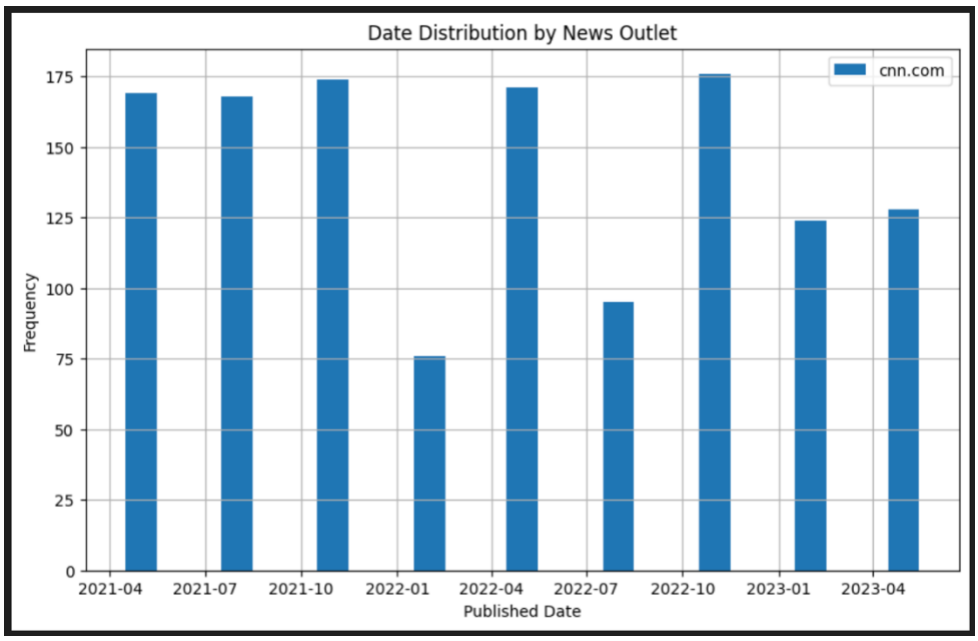
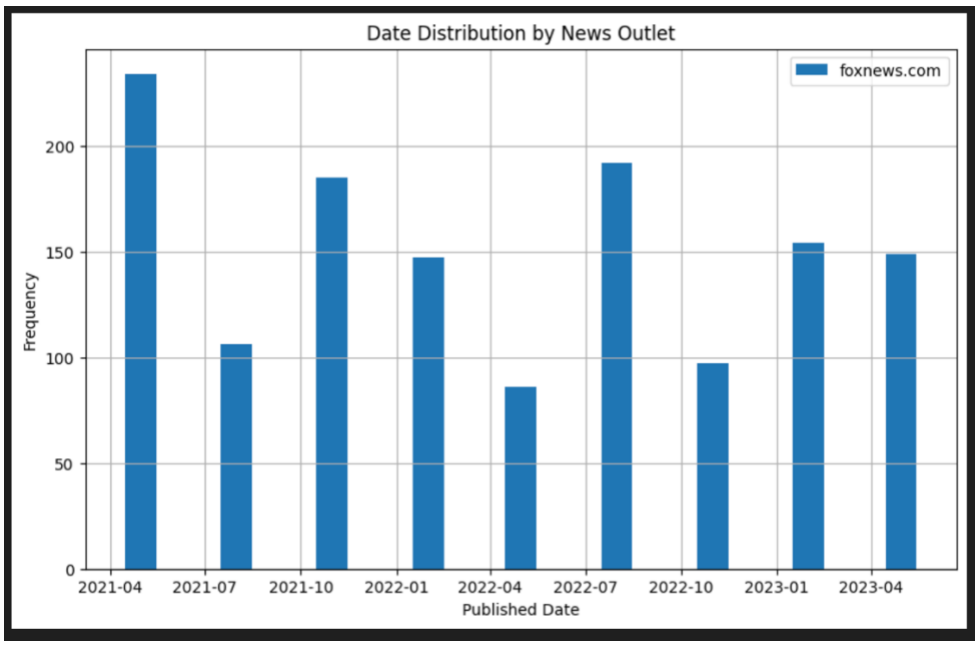


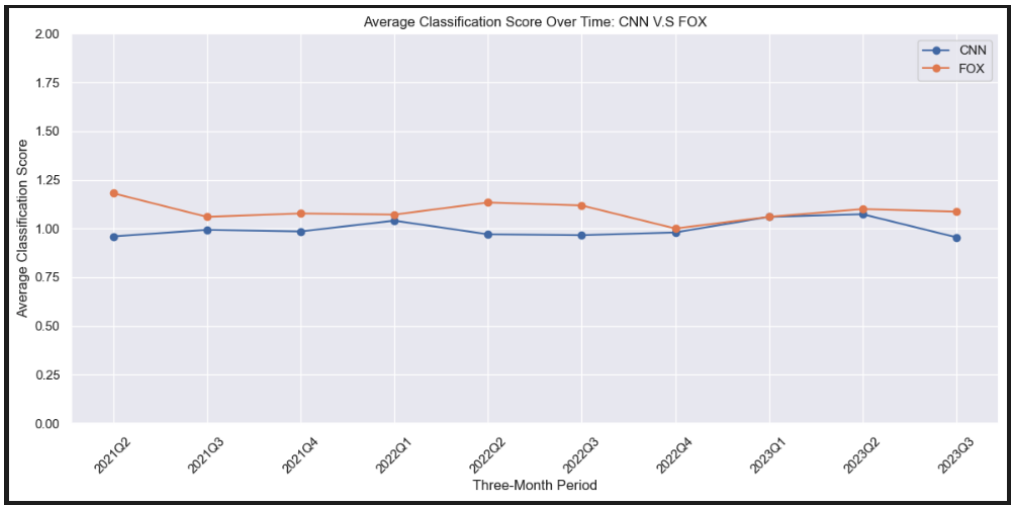
Figure 22: FOX Articles: Distribution by Period



Classifications are treated ordinally such that left-leaning articles are assigned a score of 0, center-leaning articles a score of 1, and right-leaning articles a score of 2. For each three-month interval shown above each article is classified within one of the three classes and the average of all classifications is considered as the political leaning expressed by the batch of articles of a three-month interval.

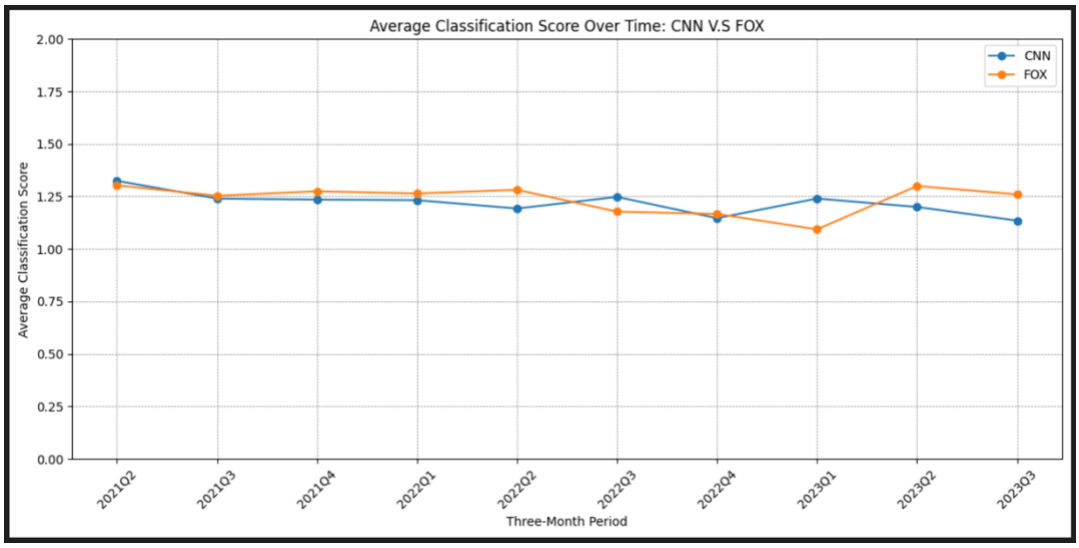
The results for the convolutional model are shown below for both FOX and CNN. As expected, CNN articles are consistently found to lean further leftward than Fox articles throughout the time interval explored. However, both groups of articles are much closer to the center of the spectrum than anticipated. This contradicts the categorization of both outlets by the academic sources initially referenced to classify the bias of the outlets used to build the training corpus. The center-oriented classification on behalf of the convolutional model implies that more textual similarities were found with articles corresponding to PBS, AP News, and News Nation Now than with outlets situated at the extremes of the spectrum. This result indicates that political expression is not the singular defining trait among news articles, thereby hindering the convolutional model's ability to isolate and focus on political bias as the primary deciding factor.

Figure 23: CNN Model: Average Classification over Time



The results for the rule-based model are shown below. Unlike the results for the CNN model, both outlets are situated at roughly the same constant position on the political spectrum throughout the interval explored. Additionally, both outlets are situated as being moderately right-leaning. While this is a reasonable result for FOX News, CNN is widely considered to express left-leaning sentiment. Once again, but to a more severe degree, this opposes the categorization of both outlets by the academic sources initially referenced to classify the bias of the outlets. The results indicate that the rule-based model does not have a viable application and is unable to correctly detect political bias in external news outlets.

Figure 24: Rule-based Model: Average Classification over Time



IV. Limitations: An Argument for Hybridization

Both models built throughout the paper are premised on the existence of a ground truth regarding the political bias expressed by a news article. This ground truth is defined by the outlet publishing the article and the academic sources found classifying the political leaning of the outlet. However, political

bias is an extremely dynamic, nuanced, and subjective form of expression that isn't fully contained within the opinions conveyed by different news outlets. While the thesis aspires to explore the presence of bias in text, both models built are ultimately designed to classify articles by finding lexical and syntactic similarities between an input string and the three corpuses considered. This is not a classification of bias, but rather a classification of text.

The rule-based sentiment detection model attempts to avoid classification criteria that stem from textual features irrelevant to an article's political leaning. It does so by limiting its attention to the sentiment expressed towards nouns. Although this approach better grasps how political bias is conveyed through news media, the model considers nouns of an apolitical character in its classification process which are in no way indicative of political leaning. Past its liberal interpretation of bias, the model primarily suffers from its viability. Given the model only interprets one feature of expression, namely sentiment expressed towards nouns, an article must contain mentions of nouns encountered in the corpus for its political leaning to have grounds on which to be classified. Additionally, if the nouns within the article that are present in the corpus are apolitical or are scarce within the corpus, the bias identified is largely unfounded.

Although the Convolutional Neural Network model boasts high classification metrics, its focus on political bias as a deciding factor in classification is unknown and seemingly limited. While the model accurately classified the three classes composing the training corpus, it concurrently identified a strong moderate leaning for two of the nation's most politically opposed news outlets (i.e. CNN and FOX). This puts into question the criteria employed by the model. However, given the nature of deep learning models, the textual features used for classification are unknown and will remain as such. The programmer is instead left to wonder what mix of features is most outstanding in the classification of articles and to what extent these features are based on the political bias expressed by each outlet examined.

An ideal approach to the classification of political bias in news media would combine the feature extraction of the rule-based model with the depth and self-correction of the convolutional model. By examining other features of text indicative of political leaning and quantifying these features (as was done for sentiment expression), an input vector suited for convolution could be assembled. Although the inner workings of the convolutional model would still be unknown to the programmer, the model's predictions would be guaranteed to only consider factors that pertain to the expression of political bias. Ultimately, by allowing the programmer to define a neural network's starting point, a series of constraints can be imposed on a model that in no way hinder its ability to self-learn but do limit the resources available to those relevant.

E. Conclusion

This thesis aims to understand and explore two opposing approaches to the classification of political sentiment in new media through a sentiment recognition rule-based model and a deep learning Convolutional Neural Network (CNN). The nuanced nature of politically biased expression in text presents a challenge in the definition of a ground truth through which to define and ultimately evaluate the two models implemented. Conscious of these constraints, a ground truth is assigned by considering where credible academic sources position widely read news outlets in the political spectrum. Furthermore, the groupings of outlets ultimately chosen land on the extreme and moderate center of the spectrum, such that the ground truth assigned to each grouping of outlets has a high likelihood of representing the bias it claims to. PBS, AP News, and News Nation Now were chosen as center outlets, Palmer Report and Bipartisan News were chosen as far-left outlets, and VDare, News Max, and Ricochet were chosen as far-right outlets.

The implementation of the CNN model is predominantly reliant on existing libraries, whereas the rule-based model is heavily detailed and demands a recognition and hand-crafted extraction of

features. The rule-based model begins with the application of conference resolution to enrich the mention of entities in the text. A POS reference algorithm is then written and applied to the text to extract the sentiment expressed toward nouns through the verbs and adjectives describing them. The sentiment expressed towards each noun within each grouping of articles is expressed through a vector, such that all groupings can map their noun sentiment onto the same n-dimensional space, where n represents the number of unique nouns throughout the three groupings. An input article is then vectorized and compared to the three vectors constructed. The deep learning model preprocesses text through the removal of stop words and tokenization. It is then given complete freedom in the recognition of patterns differentiating each class. Different layers are added to reduce the dimensionality of the input and preserve notable features.

The results show the deep learning model significantly outperforms the rule-based model in the classification of articles. The deep learning model reported an accuracy score of 89.7% compared to an accuracy of 53.6% for the rule-based model. Precision, recall, and F1 metrics were similarly higher. To explore the viability of both models, both methods are applied to the classification of articles from external news outlets, namely CNN and FOX News. This application was meant to evaluate the models on the recognition of political bias, as opposed to the classification of articles into different groupings. In tune with the initial results, the deep learning model displayed a graph that more accurately coincided with the general perception of where both outlets sided on the political spectrum.

Although a deep learning architecture is found to be both more accurate and viable than the rule-based approach, its shortcomings are visible in its application to external sources. Although predictions consistently placed CNN articles further left than FOX articles, both outlets are shown closely positioned to each other on the political spectrum. The sources used to initially determine a ground truth show both outlets significantly further away than the CNN model portrayed them to be.

The rule-based model suffered from numerous drawbacks. Most notably, its focus on a single feature through which to classify text meant that model input had to contain relevant nouns and be rich in sentiment expression for a founded classification to take place. The results indicate that this was seldom the case.

Future adjustments to the rule-based model should focus on incorporating machine learning techniques in the extraction of features and the ultimate classification of inputs. For example, as opposed to using a closest POS resolution algorithm, a decision tree could be used to distinguish between correct and incorrect references between nouns and their referencing POS. Additionally, given the scarcity of suitable input, the model would benefit from an understanding of synonymy. For example, Word2Vec could be used to assign nouns in input articles unseen in the training corpus, to their most similar noun in the vectors used for classification. Doing so would allow for a greater number of nouns to be used in classification.

Overall, this exploration delves into the extremes of techniques used for media bias classification. A shallow model with a clear aim is put up against a deep learning model with an opaque internal methodology. The rule-based model presents a strong theoretical premise but is unable to showcase convincing metrics in both testing and external applications. Both approaches suffer from shortcomings that can be prevented through hybridization.

F. **References**

Chen, W. F., Al-Khatib, K., Stein, B., & Wachsmuth, H. (2020). Detecting media bias in news articles using gaussian bias distributions. arXiv preprint arXiv:2010.10649.

Cruz, A. F., Rocha, G., & Cardoso, H. L. (2019, November). On sentence representations for propaganda detection: From handcrafted features to word embeddings. In Proceedings of the second workshop on natural language processing for internet freedom: censorship, disinformation, and propaganda (pp. 107-112).

Francisco-Javier Rodrigo-Ginés, Jorge Carrillo-de-Albornoz, Laura Plaza. (2024).

A systematic review on media bias detection: What is media bias, how it is expressed, and how to detect it, Expert Systems with Applications.

Hamborg, F., Zhukova, A., & Gipp, B. (2019, June). Automated identification of media bias by word choice and labeling in news articles. In 2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL) (pp. 196-205). IEEE.

Hube, C., & Fetahu, B. (2018, April). Detecting biased statements in wikipedia. In Companion proceedings of the the web conference 2018 (pp. 1779-1786).

Mullainathan, S., & Schleifer, A. (2002). Media bias (working paper).

Prosise, J. (2023). Text Classification with Neural Networks.

Saez-Trumper, D., Castillo, C., & Lalmas, M. (2013, October). Social media news communities: gatekeeping, coverage, and statement bias. In Proceedings of the 22nd ACM international conference on Information & Knowledge Management (pp. 1679-1684).

University of Central Oklahoma. (2022). How to Avoid Misinformation: Media Bias Chart.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.