Faculty of
Mathematics
and Informatics

VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
MODELLING AND DATA ANALYSIS
MASTER'S STUDY PROGRAMME

MEASURING POLITICAL ACTOR INFLUENCE TO MEDIA POLARIZATION FROM TELEVISION

Master's thesis

Author: Saulius Lipkevičius
VU email address: saulius.lipkevicius@mif.stud.vu.lt
Supervisor: Asist., Dr. Dmitrij Celov

Vilnius

2022

Abstract

Political polarization among Americans has grown rapidly in the last decades. It is well-known that the rise of 24-hours partisan cable news is one of the explanations for the increase. This thesis aims to determine the main polarization factors between two US primary partisan media sources – CNN and Fox News – and evaluate their central persons' influence. First, two core news sources are investigated – CNN and Fox News television shows and news articles presented on their news websites and analyzed by text properties – length, simplicity, richness, and negative sentiment level. Second, using the Poisson scaling methods Wordfish and Wordshoal, each anchors polarization is evaluated, which is their position variation on different subjects from the 2018 to 2021 period. Parallelly, by nine distinguished topics (for example, Senate, immigration, education and more), news websites articles are evaluated in the period 2016-2021. Third, the thesis contributes to polarization and text mining literature by introducing the novel data preprocessing approach and augments the polarization estimation methods incorporating anchors' status and his television shows' popularity indicators. The study suggests that CNN and Fox News are highly polarized in articles and television shows. On the one hand, there is a widening gap between the media, especially after Robert Mueller's study of Russia's influence in 2016. On the other hand, the polarization of television shows remains relatively stable. There is a tendency for CNN and Fox News anchors' polarization positions to be broadly in line with their respective media platforms. In addition, in recent years, it has become clear that formerly more neutral is getting closer to the CNN and Fox News clusters. Finally, the results obtained show that negativity level is also relevant and should be taken into consideration in the analysis of media polarization.

Keywords: CNN, Fox News, Wordfish, Wordshoal, polarization, text mining, text analysis, sentiment analysis, news articles, television shows, US.

Santrauka

Per pastaruosius 40 metų politinė poliarizacija JAV sparčiai išaugo. Vienas iš galimų to paaiškinimų — šališkų kabelinių naujienų, veikiančių visą parą, pagausėjimas. Šiame darbe siekiama nustatyti pagrindinius poliarizacijos veiksnius tarp dviejų labiausiai JAV įsitvirtinusių žiniasklaidos šaltinių — CNN ir Fox News — bei įvertinti jų centrinių asmenybių įtaką. Pirmiausia, tiriami du pagrindiniai naujienų šaltiniai — CNN ir Fox News televizijos laidos ir jų internetinėse svetainėse pateikti naujienų straipsniai, kurių analizė įgyvendinama atsižvelgiant į kiekvieno teksto savybes — ilgį, paprastumą, turtingumą ir negatyvumo lygį. Antra, naudojant Puasono mastelio keitimo metodus *Wordfish* ir *Wordshoal*, įvertinama kiekvieno CNN ir Fox News vedėjo poliarizacija, t. y. jų pozicijos skirtingais klausimais variacija 2018-2021 m. laikotarpiu. Analogiškai pagal išskirtas 9 pagrindines temas (pavyzdžiui, Senatas, imigracija, švietimas ir kt.) yra įvertinami naujienų portaluose esantys 2016-2021 m. straipsniai. Trečia, baigiamasis darbas prisideda prie poliarizacijos ir tekstų duomenų gavybos literatūros, pritaikydamas naują išankstinio duomenų apdorojimo metodą ir papildydamas poliarizacijos vertinimo metodus, įtraukiant CNN ir Fox News laidų vedėjų asmeninės įtakos ir jų vedamų televizijos laidų populiarumo rodiklius.

Analizės rezultatai rodo, kad CNN ir Fox News naujienų portaluose pateikiamos žinios ir televizijos laidos yra stipriai poliarizuotos. Viena vertus, portaluose pastebimas vis didėjantis atotrūkis tarp šių medijų, ypač po Robert Mueller tyrimo apie Rusijos įtaką 2016 m. JAV prezidento rinkimams. Antra vertus, televizijos laidų poliarizacija išlieka pakankamai stabili, taip pat pastebima tendencija, jog CNN ir Fow News vedėjų poliarizacijos pozicijos iš esmės sutampa su atitinkamų žiniasklaidos platformų pozicijomis. Be to, pastaraisiais metais išryškėjo, kad anksčiau didesniu neutralumu pasižymėję vedėjai vis labiau artėja prie CNN ir Fox News klasterių. Galiausiai, gauti rezultatai rodo, kad negatyvumo lygis taip pat yra aktualus ir turėtų būti įtrauktas analizuojant medijų poliarizaciją.

Raktažodžiai: CNN, Fox News, Wordfish, Wordshoal, poliarizacija, teksto gavyba, teksto analizė, negatyvumo analizė, naujienų straipsniai, televizijos laidos, JAV.

Contents

List of Figures

List of Tables

1 Introduction

The rise of social media and the transition to digital media has greatly influenced the traditional primary means of mass communication, including television shows and news articles. However, a median American person still spends around 347 minutes per day watching television that indicates the importance of core news outlets like Cable Network News (CNN) and Fox News (FOX) together with polarization it may create as the networks support the two main political parties in the USA – Democratic and Republican correspondingly. The recent leak of FOX anchors Hannity and Ingraham messages to Donald J. Trumps' White House chief of staff Mark Meadows related to the Capitol riot shows politics and media integrity in the United States.

While efforts are being made to study separate cases of partisan news outlets, it lacks a general overview of differences in-between media. A study [11] using content analysis shows that FOX takes a more dismissive tone than CNN, and it interviews a greater ratio of climate change doubters to believers. Second, the views of Republicans are strongly linked with the news outlet they watch, which is pro-republican, mainly FOX that gathers 97% adults who identify themselves as republican [27]. On the other hand, Democrats do not vary that much in their beliefs as a function of cable news, which suggests media with their lead anchors can influence viewers significantly.

Since 2008, Poisson scaling model *Wordfish* was introduced to study polarization phenomenon, considering mostly well-defined topics – European Parliament Speeches, Electoral Pledge in Post–War Japan, U.K. chamber of common debates. These textual data were straightforward, with specific dictionaries varying insignificantly in a number of terms and are not contaminated with opinion, commentary-based text. Hence, there is the two-stage problem: Poisson scaling models cannot extract polarization estimates when text is long, full of opinion-like text, and without a priori specified topic. Second, it does not include the text author's influence or text sentiment in the polarization estimation process. Therefore, this thesis considers how to automate text cleanse to obtain shorter, less noisy data. Also, the suggested *Wordfish* model extension takes into account anchors' status and his television shows' popularity indicators.

The second chapter, part 1 of the thesis, considers *Wordfish* and *Wordshoal* techniques and compares them with their predecessor *Wordscore*, which depends on expertly judged reference text. Part 2 presents the gathered data from media websites and television shows closed–captions together with additionally generated data such as text richness, simplicity ratio, and sentiment. Moreover, it introduces mandatory cleanse steps to prepare a text for Poisson scaling models followed by a noise reduction procedure.

The third chapter, part 1, compares CNN and FOX texts presented on websites and television shows in general on lexicon richness, text length, simplicity, and lastly, on which media is presented more negatively. Also, the relation between news sentiment and polarization was analyzed. Besides, former methods are employed to investigate the existing polarization between CNN and fox to observe its trends in the current political turbulence. Part 2 observes every media anchor position on polarization throughout 2018-2021 period in general. Additionally, quarterly measures of polarization position are obtained by employing *Worshoal* for the same period. In part 3, a method of dealing with noisy data is presented. The method helps to automate the process of data preparation to former models. In the last

part 4, the suggestive model evaluates anchors' status and his television shows' popularity indicators.

Appendix A consists of two figures for comparison of website articles polarization estimates grouped by topics, where the first one's data was preprocessed semi-manually and the second by the method presented in chapter 3, part 2. Appendices B and C presents Python scripts to retreat data from dynamical JavaScript-written pages like CNN or Fox News. Appendix D shows R script, which follows textual data preparation described in chapter 2, part 3. Appendix E displays the deep learning model Roberta implementations. Moreover, it has a function to chunk big textual data to the size of a paragraph and evaluate its sentiment. The last Appendix F represents the method suggested in chapter 3, part 2.

2 Methods

2.1 Methodology

News networks tend to reveal their political position indirectly through various content, including news portals, articles, TV broadcasts, and night shows, even if they state being unbiased. To measure partisanship or political position, we can employ the well-known Poisson scaling method *Wordscores* [20] introduced by *Laver, Benoit, and Garry* (2003). The paper mainly considers political actors on a defined dimension that is estimated by the reference document, which subject matter experts have *a priori* scrutinized for similar textual data in the same dimension. Second, the model assumes that words have the exact weights, which tells the analysts that more common words can drag estimate to a center of the analyzed dimension and that every word has the same meaning throughout different topics, which can fail to be true in the most of the cases.

When there is no reference document, unsupervised learning methods exist. Since these unsupervised learning algorithms ignore the sentence structure and context, they need data. This thesis applies the two widespread approaches – *Wordfish* [32] introduced by *Slapin & Proksch* (2008) and *Wordshoal* [19] development by *Lauderdale & Herzog* (2016), both of which directly exploit word frequencies with an assumption that words are drawn from a Poisson process, a heavily skewed distribution, as is the case of word usage. The following two sections outline and compare the methodological underpinnings of the two approaches.

2.1.1 Wordfish

Wordfish uses a bag-of-words approach:

1. A text is represented as a vector of word counts or occurrences;
2. Multiple document vectors are then put together in a term-document matrix, where:
    - each column represents a document;
    - each row represents a unique word, or term;
3. The cells of the matrix contain the number of times the unique word (term) is mentioned in each document.

The Wordfish method does not rest on documents with *ex-ante* assigned reference scores as its predecessor. Position estimates obtained by Wordfish are based only on the information in the texts and assume that the language used by political parties gives out political ideology. More specifically, Wordfish assumes that the party's relative word usage within party documents conveys information about their positions in a policy space. Besides, the model assumes that one's political position underlies how frequently one uses certain words, assuming that words are distributed from Poisson distribution.

The rate of the Poisson distribution depends on four parameters: the length of one's text, the overall frequency of the word, the underlying position, and the weight of the word in differentiating among the underlying positions [32]. The model assumes that words are generated at random and independently

known to fail on empirical grounds. However, the text classification models based on this assumption work pretty well [32]. The functionally model is described as follows:

$$w_{ijk} \sim Poisson(\mu_{ijk}), \tag{1}$$

$$\mu_{ijk} = e^{\nu_{ij}+\lambda_{jk}+\kappa_{jk}\cdot\psi_{ij}}, \tag{2}$$

where ω_{ijk} in (1) is the occurrence count of word k in media network i on topic j. It is distributed by Poisson distribution with parameter μ_{ijk} because it resembles words usage the most. Equation (2) determines relation between distribution parameter μ with coefficients $\nu, \lambda, \kappa, \psi$. ν_{ij} parameters capture the baseline rate of word j usage in a given topic media i, which controls the possibility that some topics in the analysis may have significantly longer texts than others, i.e. some media, in particular years, may write much longer stories. This helps avoid underestimation if one of the compared media covers fewer topics with more brief texts. λ_{jk} is word k fixed effect on topic j, larger for high-frequency words. It helps distinguish the case when one media may use the same word more frequently. κ_{jk} captures the importance of a specific word j in media position discrimination on topic k. A sign of κ_{jk} represents the ideological direction of the word k on topic j. At the same time, its magnitude defines the sensitivity of the word to ideological differences among article authors or anchors. Finally, ψ_{ij} retrieves media i political position on topic j. The core of the analysis will consider only former parameters ψ's and κ's that differ between both sides the most.

The same model can be applied to compare documents over time with the assumption of constant word meaning. Therefore, a document of media at the time t–1 is considered as independent and unrelated to the same media's document at the time t, which can be formulated as:

$$\omega_{ijkt} \sim Poisson(\mu_{ijkt}), \tag{3}$$

$$\mu_{ijkt} = e^{\nu_{ijt}+\lambda_{jk}+\kappa_{jk}\cdot\psi_{ijt}}. \tag{4}$$

From the former statement and equations (3), (4) it follows that the model treats every side position in year t as independent of t–1. It indicates that values at different years will be approximately the same if a network uses similar frequencies at the moment t and t–1. On the other hand, every network move is due to changes in word frequencies observed or, in fact, a policy agenda shift in texts.

The underlying position that affects word use may be linguistic style rather than ideology [14]. Seeking to remove this effect, the model requires that the word data be comparable at a minimum level over time. Therefore, using the text data to estimate the media's position over time creates an additional challenge. On the other hand, the analysts would like to use as much information in the texts as possible to estimate the position change over time. This is a critical trade-off in applying the *Wordfish* method. For instance, if the political topic changes and new vocabulary enters the political lexicon in year t, this will differentiate the texts t from those at point t–1. The procedure is discussed in section 2.2.4

At a final stage, *Poisson* distribution parameters are estimated with an iterative expectation-maximization (EM) algorithm that computes maximum likelihood estimate for a latent variable [24],

the likelihood of word mentions in each document relative to their frequency in other documents [15]. Finally, for every document, the standard error of parameters are obtained by the parametric bootstrap [9].

2.1.2 EM algorithm

Step 1: Initiating start values ψ is calculated by the logged mean count of each word. For the network fixed effect ν, we use the logged ratio of the mean word count of each media text over time relative to the first network documents in our dataset. To obtain starting values for word weights (κ) and media positions (ψ) from the word frequencies, we first subtract the starting values for the word and media fixed effects from the logged word frequencies. We then use the left– and right–singular vectors from a singular value decomposition of this matrix as starting values for ψ and κ.

Step 2: Estimate network parameters We estimate network parameters ψ and ν conditional on our word parameter expectations. In the first iteration, our expectation of those word parameters equals their starting values calculated in step 1. We maximize the following log-likelihood for each network media i at the time t:

$$\sum_{j=1}^{n}\sum_{k=1}^{m}(-\mu_{ijkt} + ln(\mu_{ijkt}) \cdot \omega_{ijkt}),$$
$$\omega_{ijkt} = e^{\nu_{ijt}+\lambda_{jk}^{start}+\kappa_{jk}^{start}\cdot\psi_{ijt}}.$$

Step 3: Estimate word parameters Based on network parameters obtained in step 2, we calculate word parameters ψ and β. For each word k, we maximize the log-likelihood:

$$\sum_{j=1}^{n}\sum_{it=1}^{l}(-\mu_{ijkt} + ln(\mu_{ijkt}) \cdot \omega_{ijkt}),$$
$$\omega_{ijkt} = e^{\nu_{ijt}^{step2}+\lambda_{jk}+\kappa_{jk}\cdot\psi_{ijt}^{step2}}.$$

Step 4: Calculate log-likelihood The log-likelihood of our model is the sum of the individual word log-likelihoods from step 3, which are themselves calculated conditional upon the party log-likelihoods from step 2:

$$\sum_{j}^{n}\sum_{k}^{m}\sum_{ij=1}^{p}(-\mu_{ijkt} + ln(\mu_{ijkt}) \cdot \omega_{ijkt}). \tag{5}$$

This flow finishes when log-likelihood (5) is less or equal to $1e$-03 or the difference between two consequative iterations (5) values differ less than $1e$-06. A user can modify both hyperparameters.

2.1.3 Wordshoal

Political texts have multiple sources of plausible variation, including language, style, topic, and preference to capture the noise created by topic variation. In this regard, one cannot assume that a word has a different meaning throughout various subjects. Further, opposition disagreement is more visible within a topic than in general word frequencies. *Wordshoal* considers former assumptions by introducing the word coefficient variation, which helps us to identify political actor position with less generalization.

Wordshoal is an extension of *Wordfish* that takes later model output as data, and the positions from a particular topic or author are scaled across groups using a second-level linear factor model. Functionally it can be displayed as follows:

$$\psi_{ikj} \sim \mathcal{N}(\alpha_j + \beta_j \Theta_{it}, \tau_{it}), \tag{6}$$

$$\Theta_{it} \sim \mathcal{N}(0, 1), \tag{7}$$

$$\alpha_j, \beta_j \sim \mathcal{N}(0, (\frac{1}{2})^2), \tag{8}$$

$$\tau_{it} \sim \mathcal{G}(1, 1).$$

This specification means that the primary dimension of word usage variation in (6), which is individual topics (ψ_{ijt}) can be more or less strongly associated with the aggregate latent dimension θ_{it} being estimated across all debates, with either positive or negative polarity for any particular topic j. If the debate is not well correlated with other debates, it gets small β. Essentially, this allows the model to select those topic-specific dimensions that reflect a common dimension – large estimated values of β,– while down-weighting the topic's contribution where the word usage variation across media or anchors seems to be idiosyncratic.

The priors on (7) and (8) coefficient β, allow the model to remain agnostic about the relative polarity of individual topic dimensions while constraining t. The study [19] has found that this method performs the best when speeches given on the same topic differ significantly, which often happens in media, especially in opposing networks. Many quantities can be calculated from model parameters.

Position on general scale. The general position of media i at the time t can be determined by quantity Θ_{it} from (6) and averaged on different levels, networks or groups of anchors.

Word loading. Association of the word compared to the general scale within all topics (it is weighted mean by frequency of word n_{kj} occurrence in each topic):

$$\frac{\sum_j n_{kj} \cdot \kappa_k \cdot \beta_j^2}{\sum_j n_j}. \tag{9}$$

The analyst seeks to estimate word-specific parameters such as (9) for each debate separately, which allows tracking whether particular words distinguish being on the left or the right, reflecting polarization phenomenon and how that phenomenon evolves across debates. Previous approaches (e.g.,

Wordfish) that combine talks into a single text assume that the association between words and positions is constant across topics or over time. Moreover, sparsity can make it difficult to measure the preferences of an author who speaks rarely. It increases the importance of assumptions about the process by which legislators choose to talk in a given debate.

2.2 Textual data collection

The most difficult challenge of textual analysis is turning complex, large-scale data into manageable information. Three-step processes have been implemented to collect data using scraping tools, generally simplify and clean not textual information found in texts, and lastly remove white noise that comes from lexicon change over time. Tiding and content analysis were run in *R* using a combination of text-mining packages, including *tm* (I. Feinerer & K. Hornik, 2008) [10], *stringr* (H. Wickham, 2010) [36] and *quanteda* (K. Benoit, 2018) [4]. Website scraping has been performed employing web browsers automation tool *Selenium* (J. Huggins, 2004) [31] and an open-source framework *Scrapy* (D. Kouzis-Loukas, 2016) [18] for extracting the data.

Selenium uses webdriver as an interface to control web pages through programming languages. So, this gives Selenium the capability to handle dynamic web pages effectively. The tool is capable of extracting data on its own. Nevertheless, Selenium cannot manage large data, but Scrapy can easily handle it. Also, Selenium is much slower compared to Scrapy. On the other hand, Scrapy cannot handle web pages that render its content using JavaScript from dynamic websites, i.e., websites that use JavaScript (React, Vue) to render content. So, the intelligent choice is to use Selenium with Scrapy to scrape dynamic web pages containing large data, consuming less time. Supplementary code can be found in Appendices B and C.

2.2.1 Data description

The empirical analysis of the thesis explores two datasets. The first consists of nine 'hard news', which is typically used to refer to timely, essential and consequential topics and include politics, international affairs, or other significant, timely events from articles in CNN and FOX news portals. More specifically, this data contains stories about ex-president Donald J. Trump, immigration, terrorism, senate, crime, education, national security, technologies, and the border wall. The second source covers television archives (https://archive.org). It incorporates so-called 'soft news' that blurs the line between information and entertainment, seven most popular TV shows anchors are considered: Carlson Tucker in 'Tonight with Tucker Carlson' (FOX), Laura Ingraham in 'Ingraham Angle' (FOX), Sean Hannity in 'The Sean Hannity show' (FOX), Bret Baier in 'Special Report with Bret Baier' (FOX), Don Lemon in 'Don Lemon Tonight' (CNN), Chris Cuomo in "Cuomo Prime Time" (CNN) and Anderson Cooper in 'Anderson Cooper 360' (CNN). Noteworthy, article data for CNN is possibly affected by selection bias. It will not reflect the actual articles' actual population since only the most relevant stories have been scraped from the CNN website because of its architecture.

2.2.2 Collection process

FOX and CNN news portals are large-scale, dynamic, JavaScript-written pages, see Appendices B and C. Hence, Scrapy with Selenium had to be employed in Python to obtain raw data of every article of interest, including articles release time, author, title, content. For TV shows, a similar script has been written to extract the date of a live show, anchor, and content of speeches of anchors from closed captions (CC). Additionally, three additional columns were created to supplement the dataset:

1. Applying fine-tuned deep learning model *Roberta* released by Google in 2018, which achieves state-of-the-art results on a range of Natural-Language-Processing (NPL) tasks, for instance, sentiment or emotion analysis to every title, we have estimated sentiment of every article title [37]. *Roberta* model has been trained on English tweets with $\sim$ 40k tweets, which generally resemble articles title type [25]. Model output values are 0 for negative sentiment, 1 – neutral sentiment, 2 – positive sentiment.

2. Type-Token Ratio (TTR), which evaluates lexicon richness or variety in vocabulary, was calculated for content per article or show. TTR measures if the target uses the same vocabulary or uses various words to communicate the idea by counting the total number of unique words (types) and dividing it by the total number of words used in the analyzed article. The closer the ratio to 1, the richer the story's lexicon.

3. Using a list of 10000 most common English words from the [17] determined by n-gram frequency analysis of Google's Trillion Word Corpus, we have compared how many words from a show or article as a percentage are from the most common words list. This measure helps to compare the complexity of a text.

2.2.3 Cleanse

Schofield and Mimno (2016) [30] have demonstrated that stemming and using other preprocessing procedures can hurt topic models. However, articles and especially television closed captions are noisy and contain many different versions of one base word, which can immensely extend the feature space of text data and create high sparsity. Hence stemming has been used to improve the performance of the textual analysis model by scaling down the feature space, assuming that we have not lost any important information by stemming, see Appendix D. Also, additional steps were employed to get the simplified text, as shown in Figure 1 (right):

1. Lowercase text, remove numbers and punctuation (?!, .), map symbols of interest, for instance, dollar sign $ to 'dollar'.

2. Remove plausible ads or references to similar articles together with repetitive information attached to a story.

3. Delete **stop words**, which are primarily neutral and do not produce political information (pronouns, conjunction, prepositions).

(CNN) Anthony Scaramucci seems to have finally learned a very, very important lesson about President Donald Trump : If you stick around long enough, he will turn on you. Always. "For the last 3 years I have fully supported this President," tweeted the former White House communications director about his onetime boss and longtime friend. "Recently he has said things that divide the country in a way that is unacceptable. So I didn't pass the 100% litmus test. Eventually he turns on everyone and soon it will be you and then the entire country."

.duval-3{width:100%;position: relative; border: 1px solid #979797; border-left: none; border-right: none;padding: 20px 0; box-sizing: border-box; -webkit-box-sizing: border-box; -moz-box-sizing: border-box; margin: auto; max-width: 660px;}
.duval-3 a{color: #1a1a1a; text-decoration: none;font-size: 0;}
.duval-3 a:hover {
color: #d9d9d9;
text-decoration: underline;
-moz-text-decoration-color: #d9d9d9;

anthoni scaramucci seem final learn import lesson presid donald trump stick around long enough turn alway last year fulli support presid tweet former white hous communic director onetim boss longtim friend recent said thing divid countri way unaccept pass litmus test eventu turn everyon entir countri

Figure 1: Text before (left) and after the cleaning (right)

4. Align date information, mostly timestamps, to the same format since new articles do not have a year stamp or are written in a different form, such as 'X days ago'.

5. Create a list with HTML, JavaScript code remainders to remove them from a text. Otherwise, the model will distinguish CNN and FOX by HTML tags.

6. Following [5] research results – the largest meaningful words found in their dataset are at most 13 letters long we remove words shorter than 3 and longer than 14 letters.

2.2.4 Noise reduction

Noise reduction is significant in the talk-like text since it is less filtered and contaminated with common words. The text scaling model can not distinguish lexicon shift from polarization change. Therefore if noise is not removed model distorts every time series of results with a positive trend compared to the 'clean' text, refer to Figure 2 (right). Figure 3 shows the same process feature space and how the cleanse removes long tails consisting of rarely used or media-specific words. The following steps were considered to solve the issue:

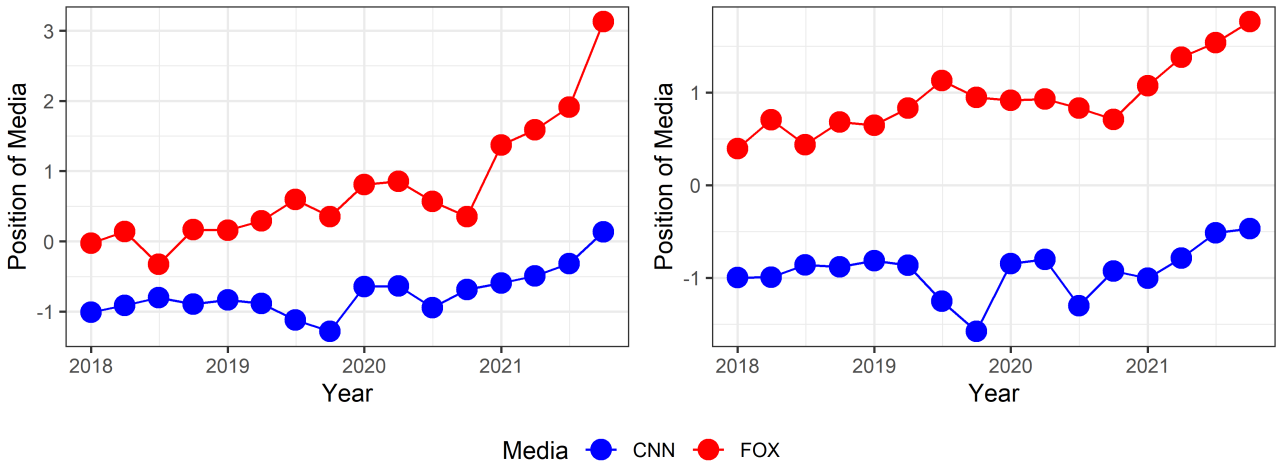

Figure 2: Polarization estimates before (left) and after (right) noise reduction

1. Delete low-frequency words to speed up the estimation process by eliminating the 'long-tail' and spelling mistakes. This step ensures that the estimation results do not hinge on the infrequently

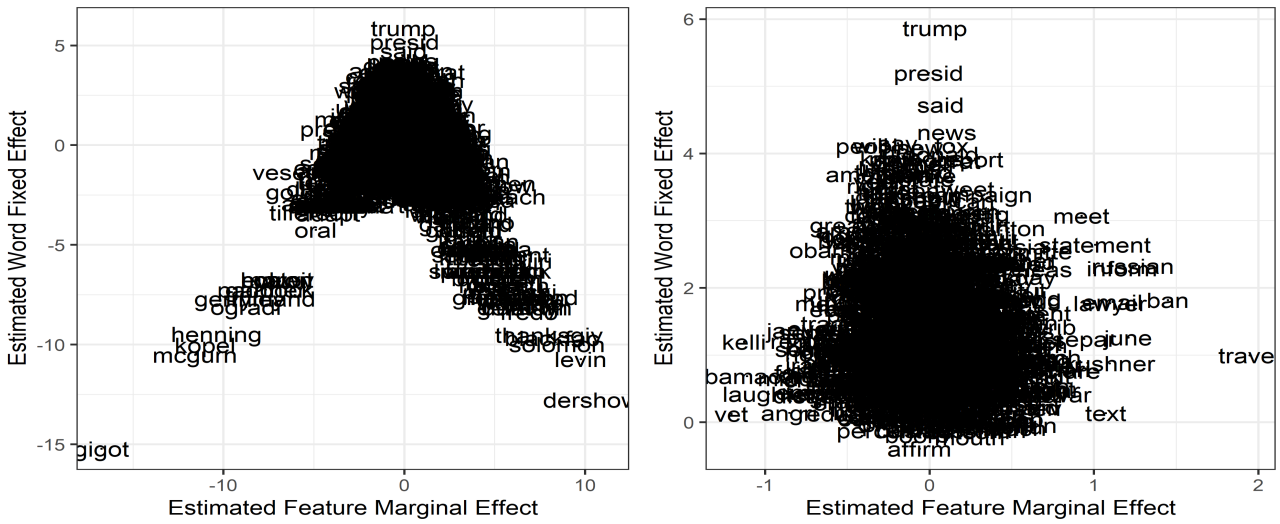

Figure 3: Words position by their fixed and marginal effect, before (left) and after (right) noise is removed

mentioned word. The minimum frequency for a word threshold cannot be high since less-used words are political side indicators that distinguish oppositions.

2. Use arbitrarily created keyword dictionaries to filter out less relevant articles to the analyzed topic or separate them into subtopics. For instance, if one examines how polarized the news about Donald J. Trump is, subtopics should be purified knowing that high-status politicians are mentioned in various topics: international affairs, environmental issues, scandals, economy, and more. In this way, we reduce feature space in specific periods that help the model not shift because of lexicon change or some subtopic signification.

3. An alternative approach to avoid political text bias in the term-document matrix is to include words mentioned in a minimum number of at least 20% of documents. Thus, one essentially keeps words that are deemed necessary enough to be mentioned over time. If the analyst does not control this fact, there would be a significant jump in all media's corresponding trend since the word usage would shift to account for the new political realities.

The final step is to inspect topics to remove redundant variation manually. For instance, Figure 4 (left) shows that CNN had a different lexicon over the immigration topic in 2016 and should be truncated because former methods did not remove the sudden wording change. Figure 4 (right) indicates that the technology subject was not polarized before the ending of 2019 most plausibly technology column announced tech innovations in general until that year. In later years many politicized events have happened: Huawei blacklisted by the US government, Cambridge Analytica scandal, 5G introduction, which are more significant in polarization analysis and create traction between the partisan media. See Appendix F for the implementation.

Table 1 presents that noise reduction helps remove remaining HTML or JavaScript code parts that have significant value determining which side media leans to. For instance, 'Right' word 'click' before noise removal has a high positive value of 5.62. The cleanse helps signify other truly relevant words;

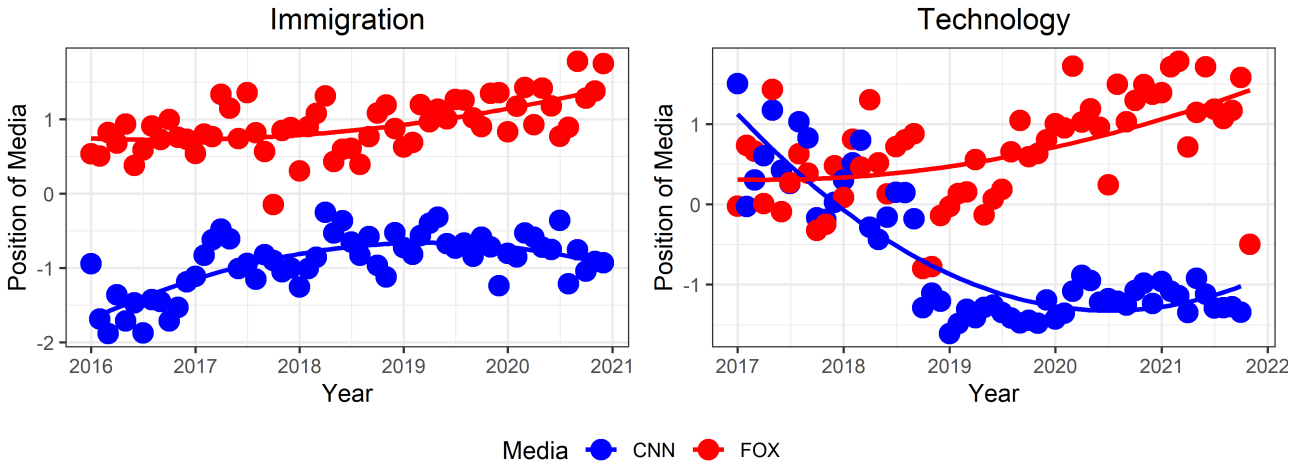

Figure 4: Examples of lexicon shift (left) and no polarization period (right)

'impeach' goes from a value of 9.98 to 11.35. However, on average, words significance decrease.

Table 1: Most significant words and their impact to distinguish polarization position before and after noise is removed. Values are calculated by formula (9)

Right Word	Impact	Left Word	Impact	Right Word (no noise)	Impact	Left Word (no noise)	Impact
impeach	9.970638	photo	-7.979476	impeach	11.345899	caption	-6.064080
biden	6.889392	hide	-7.009273	ukrain	6.786785	comey	-4.791263
democrat	6.065750	caption	-6.231234	fox	6.539154	inaugur	-4.438929
ukrain	6.063482	trump	-5.364353	mueller	5.060476	trump	-3.895722
click	5.617223	inaugur	-5.050068	will	5.051441	russia	-3.514897
news	4.990321	russia	-4.811174	ukrainian	4.162436	spicer	-2.798298
pelosi	4.769571	obama	-4.555452	deeper	3.651247	administr	-2.690250

2.3 Anchor influence data

Television show presenters can act more or less significantly on the general polarization. It depends on the number of viewers, anchor reputation or trust, and show's narrative structure. Data were gathered from multiple sources:

1. "The Hollywood Reporter" released a study on the trust of various news anchors, which gives us information about which anchor is trusted fully and which one only partially [34].

2. Article [12] presents two measures: a total audience for the week and number of viewers in age group 25-54.

3. Additionally, using the deep learning model *Roberta*, see Appendix E, the negative sentiment of every show has been calculated, which shows what proportion of chunks of 100 words are negative.

Most measures do not have consistent measures throughout the analyzed period 2018-2021, mostly only in 2020. Hence, every data entry with a missing value was filled with a previous or following not null value.

3 Results

3.1 General CNN and FOX comparison

3.1.1 Text properties

According to the report [28], Fox News has 5% more viewers with High School or lesser degree and 5% fewer College grads or higher following the median age of viewers of 61 years old at the ending of 2015 - the same as CNN. It is natural to shift towards your audience with shorter, more straightforward, opinionated news. The top part of Figure 5 indicates that the CNN news website has a negative trend in article-length after the Presidential election of 2016 following the period of relaxation until the next election primaries in mid 2019 which gives a spur for political news. The opposite reaction of CNN media behaviour in the post 2020 election period can be explained by the *COVID-19* pandemic's impact on the media system and news consumption during its first outbreak [7]. On the other hand, FOX has had steady positive news coverage growth over the observed years on the television and media website. The development can be explained by how extra coverage impacts United States politics – FOX is substantially better at influencing Democrats than CNN is at influencing Republicans. Also, it is known that CNN's focus on centrists is more negligible than Fox's. [22].

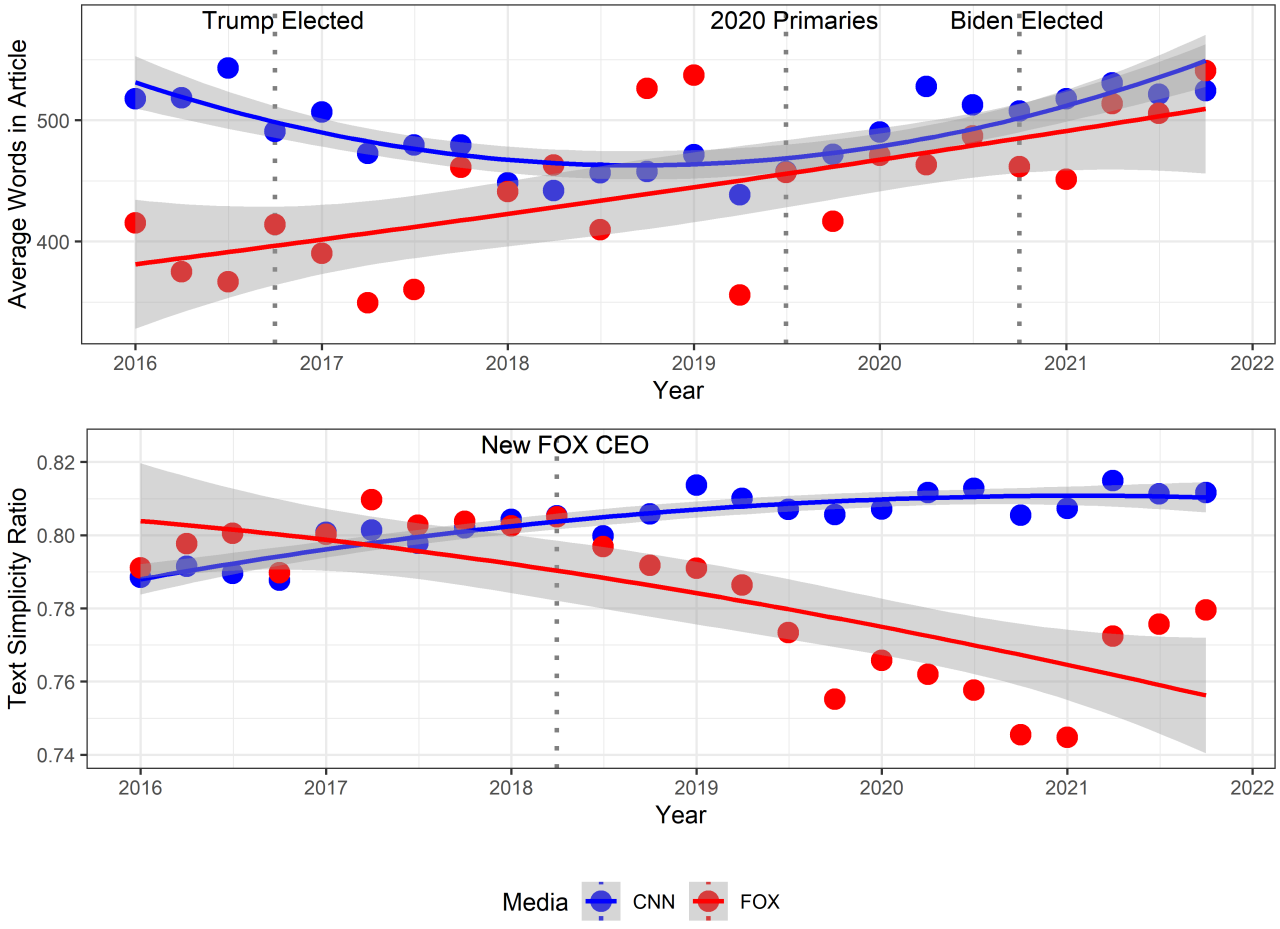

Figure 5: Average article length (top) and simplicity ratio (bottom)

Considering the bottom part of Figure 5, there is a negative trend with a sudden drop after the 2018 first quarter in text simplicity of FOX articles. Also, CNN has a minor constant increase at the same time. Fox News trend can be linked with the new chief executive officer Suzanne Scott who, by Washington attorney Robert Barnett runs Fox News as more of a business than as a political machine [21].

Media also provides television shows focusing on commentary and opinion-style talks on the coverage of the important events live. Television blurs the line between information and entertainment, creating white noise that is not always cleaned by methods mentioned in section 2.2.3. Keeping that in mind, a talk in a show should be more straightforward. Figure 6 (top) indicates that no noteworthy difference in shows length whatsoever. However, as shown in Figure 6 (bottom), CNN text simplicity is higher by one per cent, more specifically, it varies around 0.825 and Fox News at 0.815, which does not differ significantly from article text in websites of respective media.

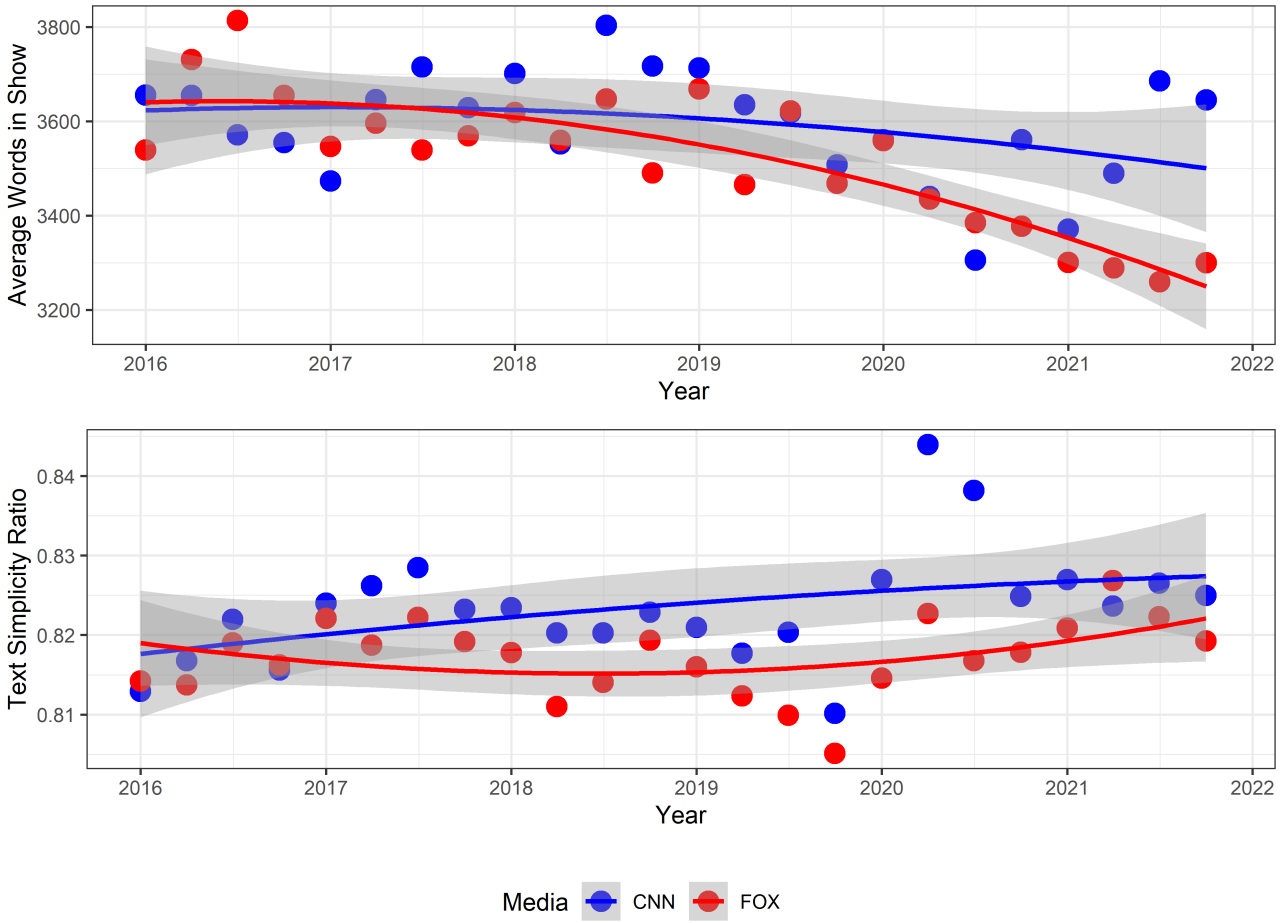

Figure 6: Average television show text length (top) and simplicity ratio (bottom)

In addition to the text simplicity and length measures, *TTR* values are presented in Figure 7. The graph on the left shows immense lexicon richness polarity in the period from 2016 to 2019, strengthened by Fox News' having met CNN's average article length around 2019. TTR measure is sensitive to the number of words used in a text; hence the gap between FOX and CNN can be explained by the difference

in article structure and purpose presented in a media outlet, where FOX prefer summaries and link to one of their television shows that are considering the similar issue, refer to Figure 5 (top). CNN is more descriptive, although the mentioned dissimilarity disappears post 2019, so we can conclude that FOX has a richer lexicon in articles and television in general, especially a post 2019, refer to Figure 7 (right).

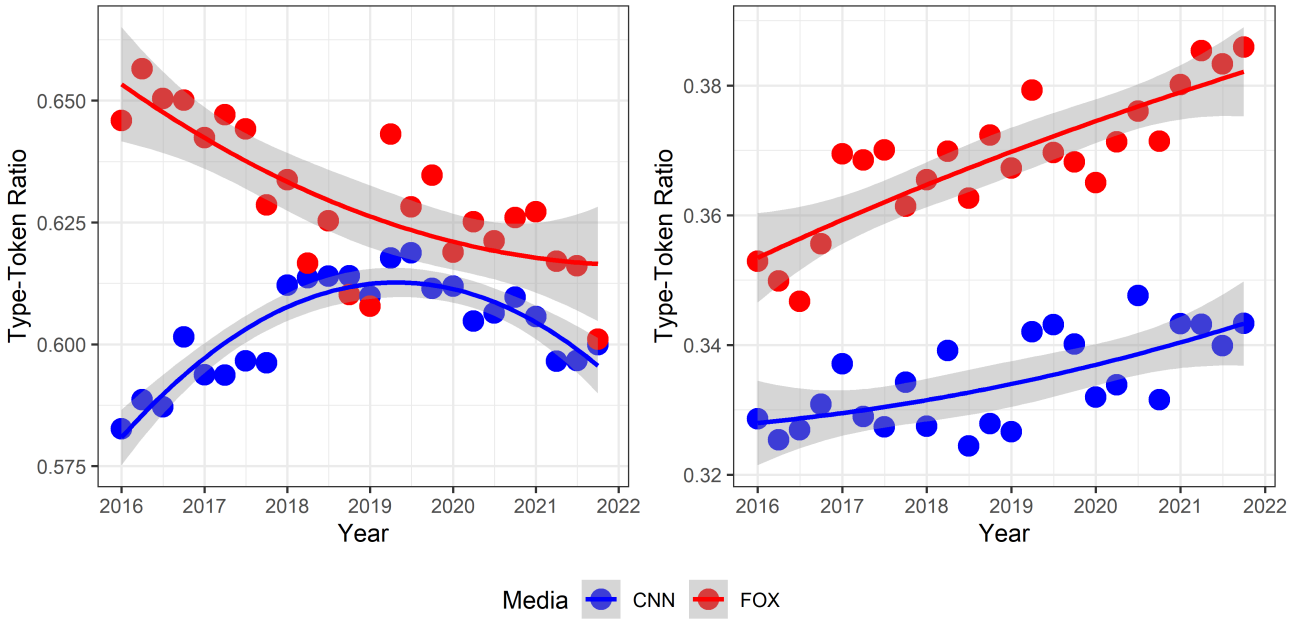

Figure 7: Lexicon richness measured by TTR in websites (left) and television (right)

3.1.2 Sentiment analysis

The experiment by researchers *Marc Trussler* and *Stuart Soroka* have shown that participants often choose stories with a negative tone – corruption, setbacks, hypocrisy, and so on rather than neutral or positive stories. Additionally, people who were more interested in current affairs and politics were particularly likely to choose bad news – this phenomenon is called *negativity bias* [29]. Based on this assumption, negative news should be emphasized in studies of polarization.

Figure 8 (top) displays a notable difference in the overall ratio of negative headlines of articles which is approximately 45% on average in 2016 later strongly fluctuates from 35% to 49% following a rapid rise after 2020 primaries from of 39% till 50% at the end of the sample. However, recent changes in negative sentiment for Fox News can not be addressed only to changes in the White House. Oppositely, CNN behaves more stable trough 2016 – mid 2019 and has around 41% of articles with negative headlines following the consistent decrease in negative headlines up to the end of the sample. To conclude, FOX news is more likely to be negative on average than CNN.

Given that these stories and shows often negatively cast the opposing political party, those who consume their news from these sources are the most likely to develop and maintain negative attitudes toward the counter-party. Partisanship also has a powerful influence on America's opinions of political leaders, including the president. According to recent data from the Pew Research Center, there is

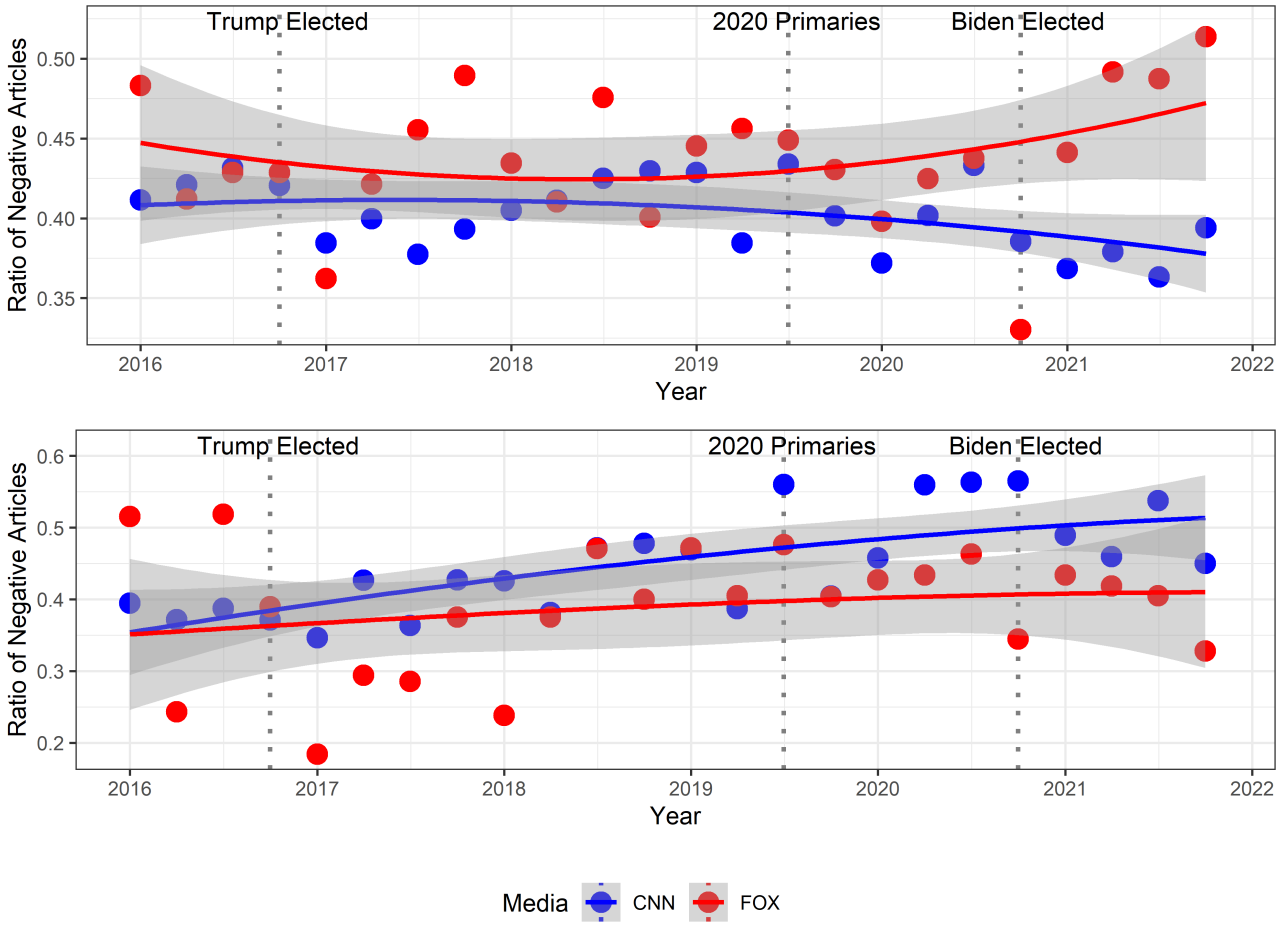

Figure 8: Ratio of negatively titled news article for all subjects (top) and if the subject is Donald J. Trump (bottom)

a growing partisan divide over media preference, CNN is distrusted by 58% of respondents in 2009 compared to the 33% in 2014, Fox News has 13% and 19%, respectively in 2014 and 2019 [26]. Indeed, among Trump-supporting Republicans during the 2016 GOP presidential primaries, 44% listed Fox News as their primary news source. In contrast, only 4% of Democrats who supported Hillary Clinton during the Democratic primaries listed Fox News as their primary news source[2]. Study [3] explores that states that partisan medias "defend" their political leaders, which can be observed in Figure 8 (bottom), CNN increasingly develops more negative narrative, compared to the average seen in Figure 8 (top) in subjects related to Donald J. Trump. On the other hand, Fox News demonstrates the opposite tendency and keeps their news related to Trump less negative on average than in general. However, period post 2020 cannot be evaluated directly because of significant events that shook the world and added white noise – the Senate acquitted President Donald J. Trump of impeachment charges, The United States made moves to leave Afghanistan, George Floyd's death sparked global protests, and lastly *COVID-19* pandemic rise.

3.1.3 Polarization

Fox News has been established with the slogan "Fair and balanced" and CNN "CNN, the most trusted name in name in the news", which should indicate their unbiasedness and fairness. The objectivity of hard news that includes every topic analyzed is rejected by observer polarization within these topics see Figure 19 in Appendix A. Topics are compared to well-known polarized political actor Donald J. Trump, a perfect example of Democrats and Republicans increasing division in their policy preferences and feelings about the parties and their leaders [1]. To conclude, CNN and FOX news portals are heavily polarized in every topic the data covers.

To analyze the general trend of polarization *Wordshoal* has to be employed since word meaning can differ from subject to subject. Figure 9 reveals that from 2017 there is a slight decrease in polarization until 2018 following repose until the influential Mullers report, which was followed by the first impeachment of Donald J. Trump where he was accused of withholding military aid, and an invitation to the White House to Ukrainian president Volodymyr Zelensky to influence Ukraine to announce an investigation into Trump's political opponent Joe Biden and to promote a discredited conspiracy theory that Ukraine, not Russia, was behind interference in the 2016 presidential election [33]. As can be seen, Fox News had a steady drift from CNN, which was stimulated by the beginning of the impeachment and continued until the middle of the impeachment. On the other hand, CNN contributed to the polarization rise by more steady movement from Fox News.

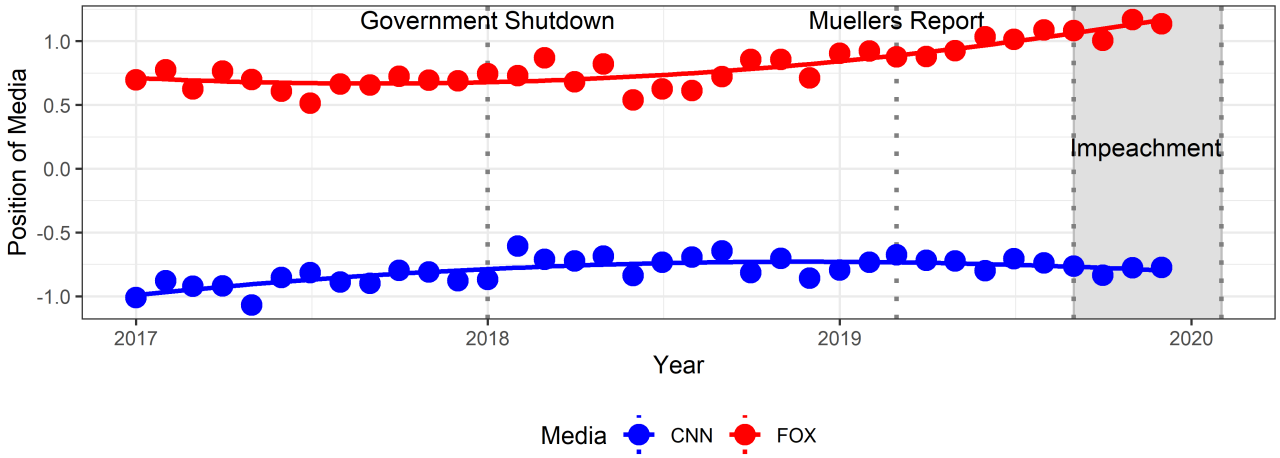

Figure 9: Media websites polarization estimate using the Wordshoal method. All topics included

3.1.4 Sentiment polarization

In section 3.1.2, it is observed that media is tilted to their leaders by sentiment. Following the same procedure as in section 3.1.3, polarization is calculated, separating media text by an article title's sentiment. Therefore, four-time series are observed: FOX – negative, CNN – negative, CNN – neutral, and Fox – neutral news. Figure 10 indicates a significant difference between negative and neutral news CNN present in their website, as well as FOX. However, it is less dissimilar compared to CNN. Additionally, we can observe that FOX negative news is more polarized than the CNN neutral. Moreover, CNN negative stories use words more similar to Fox News.

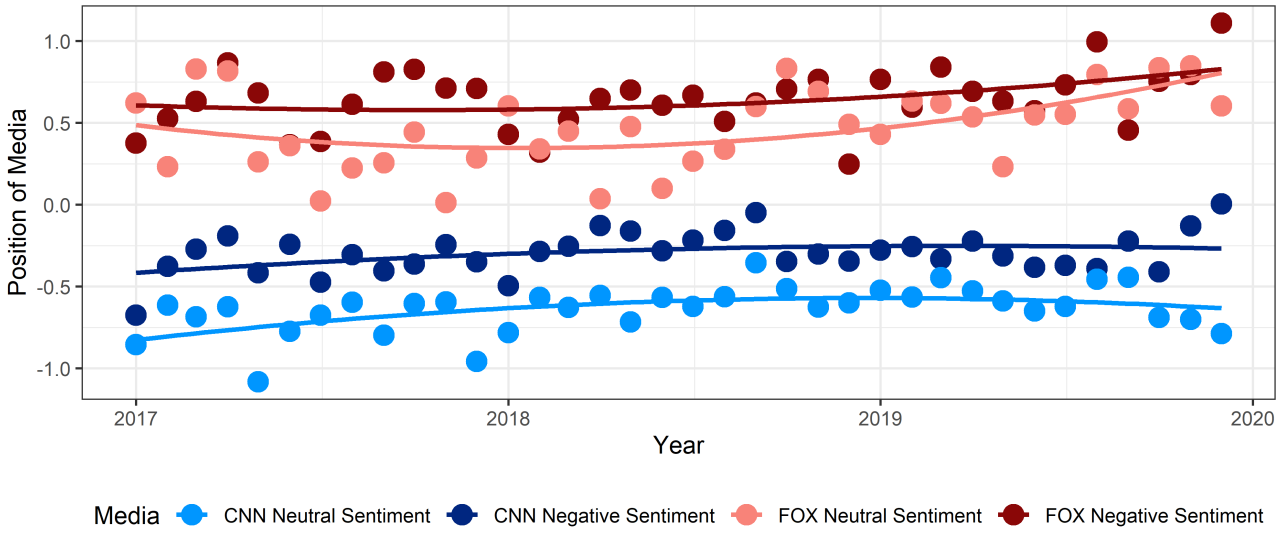

Figure 10: Media websites polarization estimate where articles are grouped by their title sentiment

3.2 Television anchors analysis

TV news anchors have a long history of being trusted sources for world news and events. CNN and FOX have shaped networks by emphasizing personalities since 2013 and 2002. As a consequence, the media provide more commentary and opinion shows nowadays [6]. The best news anchors do not just report the news. They analyze current events and put them into perspective, often having a profound effect on politics and public opinion.

3.2.1 Anchor political position

Tucker Carlson has campaigned for Republicans and Republican-affiliated causes during his time as a Democrat, which indicates his political position in the right even though he belonged to the Democrat party until 2020 [23]. Laura Ingraham and Sean Hannity belong to the Republican party, joined Fox News in 2008 and 1996, respectively. They continue to work there in conservative political commentary shows "Hannity" and "Ingraham Angle". Exceptionally, only Brian Baier has cultivated an image as the consummate down-the-middle, straight-news anchor, and those who watch the channel question his impartiality in this polarized media landscape.

Analyzed CNN anchors are independent and apolitical towards presenting news as they claim. Nevertheless, a political position can be seen from open statements. Anderson Cooper was criticized for not more adamantly pursuing a question regarding Hunter Biden during the Democratic presidential debates. Hence, many believe Cooper's opinion is skewed left. Chris Cuomo has shown benevolence to neither side – criticized the progress the Democratic party was making at that time. On the other hand, during a 2019 interview with Donald J. Trump supporter Kellyanne Conway, Cuomo criticized the former President's lack of care for the American people. Lastly, Don Lemon's political stance resembles the rest of CNN anchors.

Wordshoal method has been employed to measure polarization in television between the most popular TV shows, CNN and FOX. Results in Figure 11 confirm the clear difference between CNN and

FOX's most popular television shows, which concentrates overtime to two groups, Hannity, Ingraham, and Tucker at the right and Lemon, Cooper, Cuomo at the left together with Fox anchor Baier in between. His position fluctuates between the right and the left and settles with FOX anchors, which indicates that Baier truly is the down-the-middle, straight-news anchor. However, an additional assumption had to be made – the model considers any quarter as a debate in which the exact wording can have a different meaning because there is no other way to determine the main subject in TV shows at a specific time.

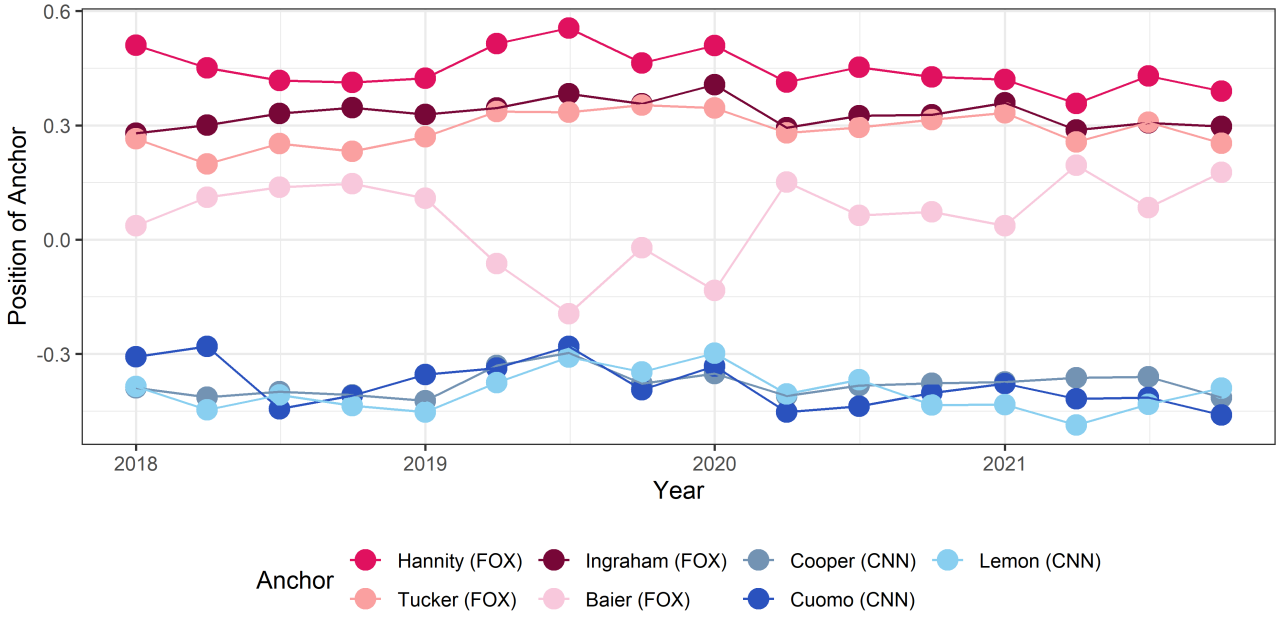

Figure 11: Anchors individual polarization position by quarters from 2018 to 2021

Figure 12 supplements this analysis by considering general anchor position throughout the 2018-2021 period. CNN presenters grouped at the far left, Fox News anchors at the far right, and Hannity was the most distant. Also, as observed in Figure 11, Baier is almost in the middle of FOX and CNN.

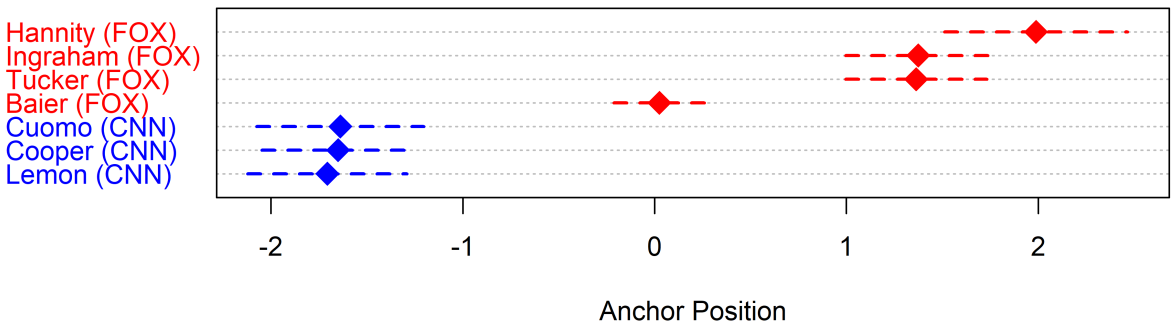

Figure 12: Generalized polarization position of anchor position in period from 2018 – 2021

3.2.2 Lexicon analysis

It has been observed in section 3.1.1 that Fox News media presents with a richer lexicon using less common words. The complexity of subject matter also affect the type of topic, generally, more complex topics require a more complex and diverse lexicon [35]. The complexity of the subject matter also affects the type of topic. Hence, every FOX anchor has a richer lexicon or discusses a more complex subject, possibly caused by deviation from a main event of a show with other affairs, refer to Figure 13. Therefore, CNN presenters give more straightforward news concentrating on fewer issues at once.

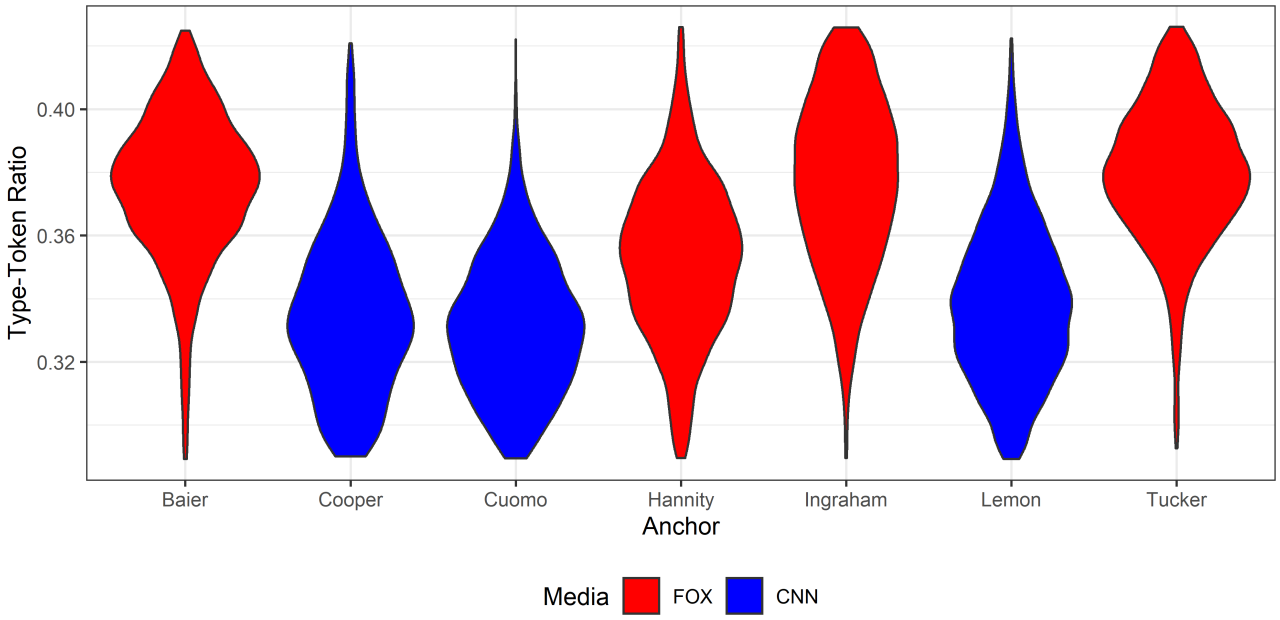

Figure 13: Anchors lexicon richness by the TTR estimate

3.3 Dealing with noisy data

First-level scaling in the *Wordshoal* algorithm is based on running the *Wordfish* on every document group before using its output as input to the factor analysis. Only two steps are considered for processing data in the first-level stage:

1. No groups or author partitions are a single row, which means there can not be a single case for the following groups.

2. Terms that were not used at least twice in any document.

However, the former steps are enough only on strictly written political manifests, which do not have prior noise as television shows do. Second, some topics might not be politicized and give no information to polarization analysis whatsoever. Therefore, an alternative solution how to remove white noise is described below.

Poisson scaling models are sensitive to a shift in lexicon over time and, with immense change, lose the ability to measure polarization. Figure 14 (left) shows the example of noisy data fitted in

the *Wordfish* model. It can be seen that there is a positive trend over time with high variability and amplitude in polarization measures.

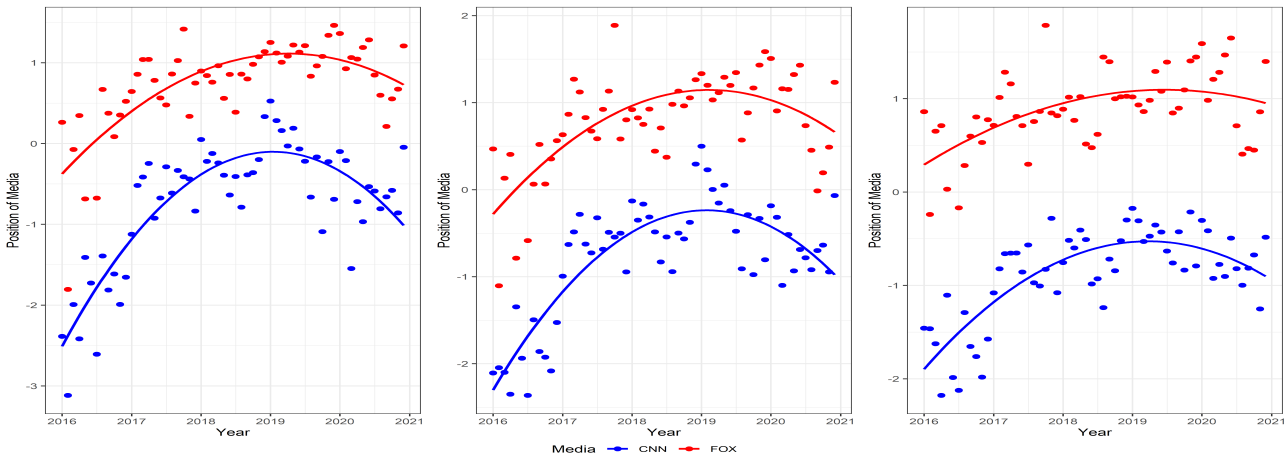

Figure 14: Polarization estimates by *Wordfish* when $p = 0.2$ (left), $p = 0.6$ (middle) and $p = 0.92$ (right)

Let define p as a threshold for a word term to appear in the model as a ratio in how many documents or debates it appears over the tally of documents. For instance, $p = 0.6$ would indicate that every term used in at least 60% of debates should be considered.

First of all, define the initial p value. It is suggested to have at least 0.2 then process steps defined below are used to find final p:

1. Clean less used terms that p value indicates and run the *Wordfish* on the remaining data.

2. Check the ratio of values that crosses horizontal value 0 to the opposite media side. If it is less than 0.05, run one more iteration to check for possible improvement. If there is no increase in p, move to the next step. Otherwise, increase p by a constant of prior choice and continue the loop until p is not growing, then move to step 3.

3. After no more outliers can be removed by step 2, variance is observed of current polarization time series and compared with the following iteration polarization variance to detect if the process can be stabilized a bit more. If variance decreases at least by 0.01, increase the coefficient p, else stop the loop.

4. Last, linear regression is applied to the whole dataset. The growth coefficient of regression helps identify a remaining positive growth in the polarization time series and how significant it is.

Algorithms' impact on the noise is displayed in Figure 14 graphs, where the left graph has the initial $p = 0.4$ value. The middle graph shows how the coefficient pushes sides apart by being increased. The right graph with the final $p = 0.92$ value has a significantly smaller variance than the former. Step 2 restricts a constant growth in the position. Step 3 helps to obtain less fluctuating polarization estimation. The amplitude of the position of media varies in two times shorter ranges. The last step assures the user with no constant positive growth (more significant than 0.1) in the polarization time

series. If the growth exists, the user is notified to review the topic and split it into subtopics to remove additional white noise. Table 2 compares coefficient p values when chosen manually by observing changes empirically, when only the first two steps are used in the method, and when the model is used fully. Also, The last column presents topics that were marked by a remaining positive trend, hence a user should revise it.

Table 2: Coefficient p choice by manually observing a topic versus coefficient p suggested by the method at various steps

Topic	Manually Selected	After Step 2	After Step 3	Positive Trend
Trump	1	0.96	0.98	1
Border Wall	0.95	0.8	0.92	1
Crime	0.9	0.88	0.92	0
Education	0.9	0.8	0.8	1
Immigration	0.95	0.8	0.86	0
National Security	0.95	0.9	0.92	1
Senate	0.95	0.92	0.96	1
Terrorism	0.9	0.88	0.92	0
Tech	0.95	0.9	0.96	0

Television shows and news websites dynamically change in concept over time. For instance, they signify a topic with more in-depth analysis, spend more resources on it, or see the new potential after some significant event. This creates an impulse in the polarization position of media, which can be observed in Figure 14 (right) shows the case of polarization of the Senate topic, which displays the mean shift in CNN's polarization that indicates 2016 year data redundancy. Post reduction graph can be observed in Figure 15. Therefore, continuing with the algorithm, the next step after p value

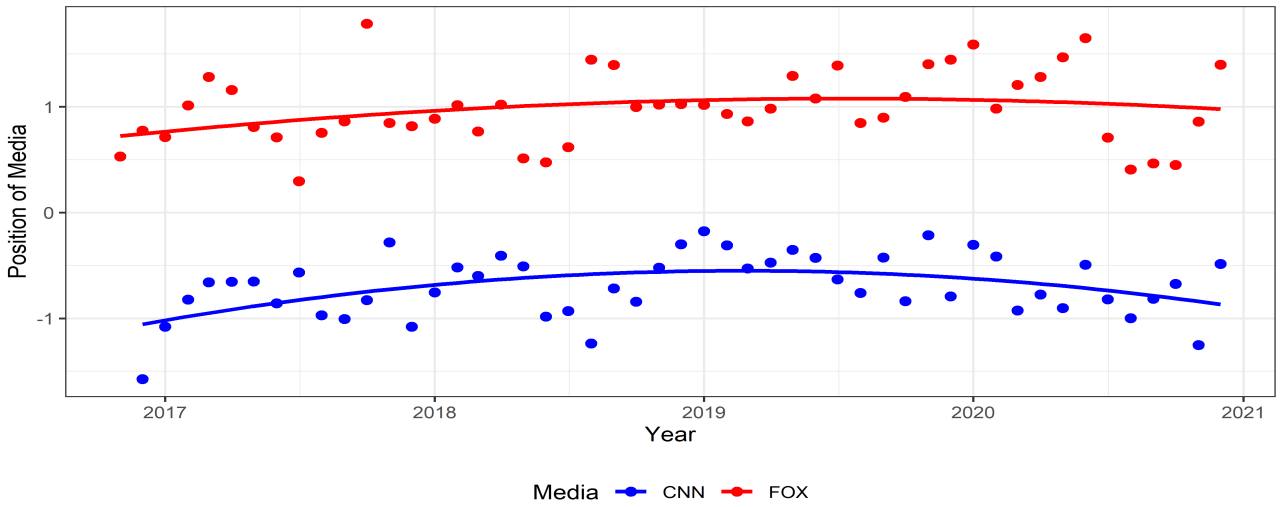

Figure 15: Clean polarization estimation after the significant mean change period is removed

has converged is the change-point analysis must be applied to delete redundant data responsible for additional noise in the generalized polarization. R package *changepoint* function meanvar have been chosen, which calculates the optimal positioning and (potentially) number of change–points for data mean and variance using the user-specified method described in [8]. If at least one of the media's polarization shifts significantly in mean or variance appeared and a change–the point is at the beginning

of the time–series, the period is deleted. Implementation is displayed in Appendix F.

The same analysis is applied when the algorithm does not help p to converge. This issue occurs when a topic has not been polarized, or the media did not focus on that topic– all values fluctuate around value 0 or at one of the sides, which can be seen in Figure 16.

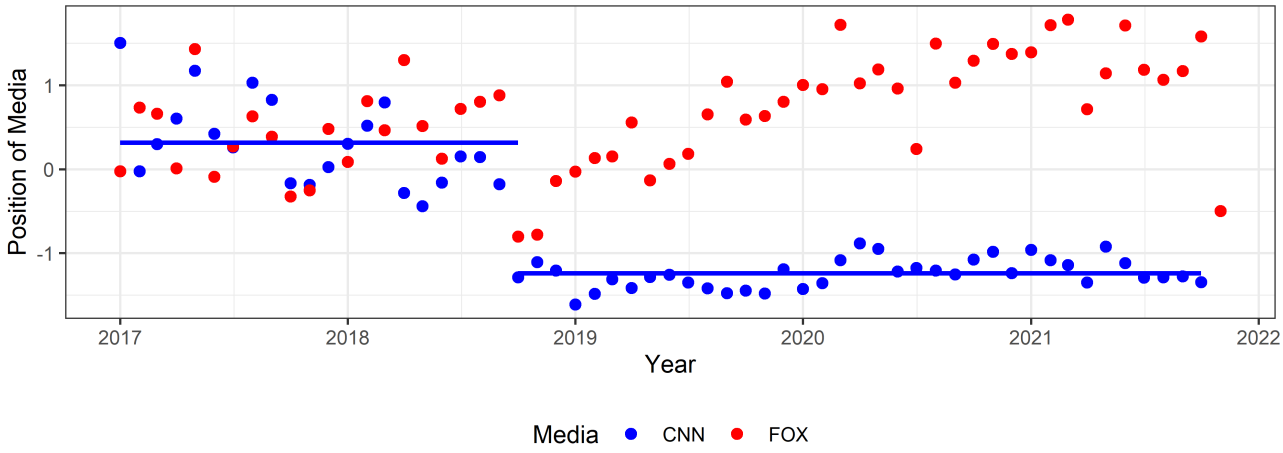

Figure 16: Polarization change–point analysis to remove not polarized period

The method has been applied to the former website articles from section 2.2.1 to confirm its validity, see Appendix A, Figure 20. The manual topic polarization cleanse does not differ significantly from the cleanse suggested by the model, see Appendix A, Figure 19, where "After Step 2", Table 2, values are chosen as p. Thus, even a simpler version of the method is sufficient.

3.4 Measure influence

The model described in this section tries to evaluate political actors, more specifically television show anchors, influence on the polarization between media and obtain polarization position shifted by anchors' status and his shows' popularity indicators.

3.4.1 Model to measure influence

Television shows or their anchor relevance can be measured by various estimates in different units, such as thousands of viewers and the percentage of the audience that trusts an anchor. Let us denote x_{itkm} as anchors i at the time t, k measure of interest that can give information about anchors influence and m that determines what side anchor belongs to:

$$y_{itkm} = \frac{x_{itkm} - min(\mathbf{x}_{km})}{max(\mathbf{x}_{km}) - min(\mathbf{x}_{km})} \quad \forall \quad 1 \leq i \leq I, \quad 1 \leq t \leq T, \quad 1 \leq k \leq K \quad \text{and} \quad 1 \leq m \leq M, \quad (10)$$

$$\text{where } \mathbf{x}_{km} = (x_{11km}, x_{21km}, \ldots, x_{I1km}, x_{12km}, \ldots, x_{ITkm}).$$

Therefore values are *unity-based normalized* by formula (10) to bring all values into the range $[0, 1]$. Also, it signifies values that are way higher than the minimum.

Next, every measure should be assigned a weight to identify which data is more significant to the model:

$$z_{itm} = \sum_{k=1}^{K} \alpha_k y_{itkm} \quad \forall \quad 1 \leq i \leq I, \quad 1 \leq t \leq T \quad \text{and} \quad 1 \leq m \leq M. \tag{11}$$

Every measure k is assigned weight α by formula (11), which indicates every estimate's significance. A researcher can assume that, for instance, audience size is more important than how negative news is. Therefore, audience size estimates will obtain a more immense weight for it:

Separation by media assures that no actor from one partisan side can influence other actors influence measures, and we can find core figures on both separately. Consider $\mathbf{z}_{tm} = (z_{1tm}, z_{2tm}, ..., z_{atm})$, where i is anchor at the time t from media m:

$$\sigma(\mathbf{z}_{tm})_i = \frac{e^{\beta z_{itm}}}{\sum_{j=1}^{K} e^{\beta z_{itm}}} \quad \forall \quad 1 \leq i \leq I, \quad 1 \leq t \leq T \quad \text{and} \quad 1 \leq m \leq M. \tag{12}$$

The set is filled into *softmax* function (12), so a sum of presenters influence from one media sums up to one and influence distributes in percentage type of ratio. Also, base coefficient β can be higher if a user intends to signify anchors with bigger components input influence.

Let $\sigma(\mathbf{z}_{tm})_i$ determines actors i influence to the general polarization of the media m at the time t. Therefore, media anchors estimated position at the time t from the *Wordfish* multiplied by the influence and summed up in their media m results in general polarization:

$$\tilde{\theta}_{tm} = \sum_{i=1}^{I} \sigma(z_{tm})_i \cdot \theta_{itm} \quad \forall \quad 1 \leq t \leq T \quad \text{and} \quad 1 \leq m \leq M. \tag{13}$$

3.4.2 The model on CNN and FOX data

The model defined in section 3.4.1 was tested by supplementing the anchors' polarization estimates dataset from section 3.2.1 (see Figure 11), which were obtained by employing *Wordfish*, with the dataset from section 2.3. The model hyperparameters in (11) for weights $\alpha = (5, 1, 1, 10)$, where weights are for negativity level, complete trust, partial trust and a number of viewers data, respectively. It indicates that the main influence source is viewers number, then twice smaller value gets negativity level, and trusts have a small influence. Moreover, $\beta = 2$ in (12) was set to signify more distinguishable anchors.

Comparing results presented in Figure 17 to Figure 11, the CNN trend did not shift significantly because every anchor position has been similar to the clusters. Also, it can be seen that neither of the presenters has a strong influence, except Lemon in 2020 $Q2$-$Q3$, Figure 18 (left). The only phenomenon observed is that Cuomo has a surge in influence after 2020 $Q2$. However, Cuomo's polarization is integrated with CNNs, and no effect is seen in Figure 17. Interestingly, Fox News has Hannity, who deviated from the FOX polarization cluster and Braier that fluctuated between CNNs and FOX clusters, Figure 11. Therefore, if one of them has more significant influence, they can push the media polarization towards their own position. However, Figure 18 (right) indicates that Baier's influence is minimal, mainly because his viewer's number is relatively small. On the other hand, Hannity has had a more decisive influence up to 2020 but lost it to Tucker, who increased his power

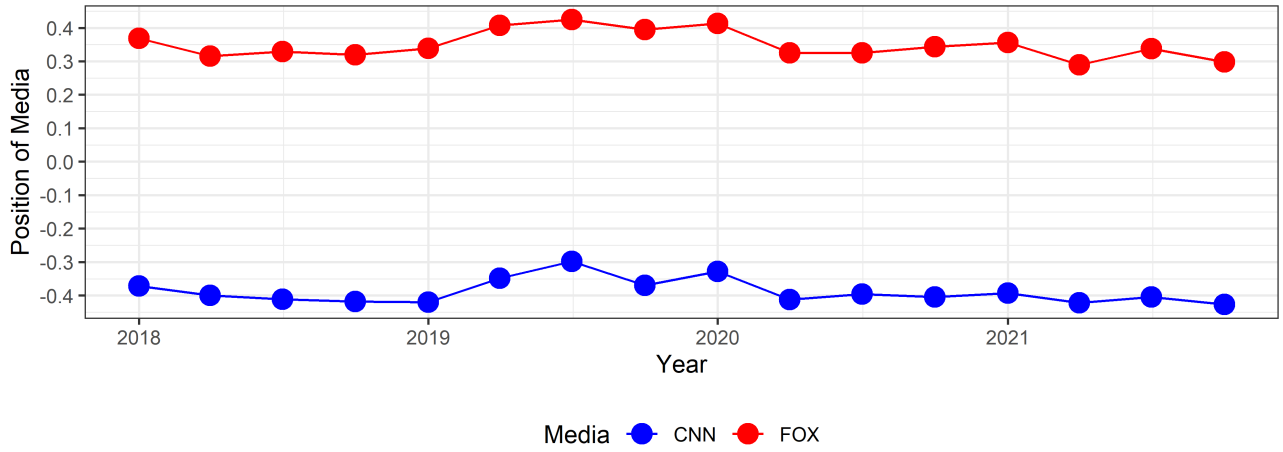

Figure 17: Media polarization with social factors included

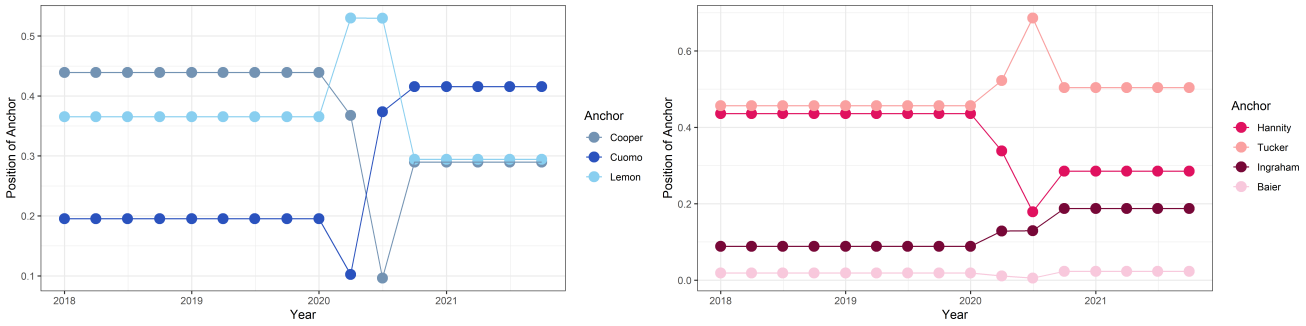

Figure 18: Anchor influence to the polarization with social factors included

with the tremendous number of viewers of his shows "Tucker Carlson Tonight" [13]. Therefore, Tucker normalizes FOX polarization around his own.

4 Conclusions and discussion

As traditional media is still central to people's everyday lives, it is important to understand how media polarization shifts in the current turbulent political environment. It was observed that website articles and television show textual data require additional cleaning steps to fit Poisson scaling models because lexicon shift distorts the model output. Also, it requires at least a semi-automated solution to remove white noise or not polarized periods if many topics are considered in research. Next, the CNN and FOX comparison helped us empirically refute the existing claim that Fox News presents in a simpler, shorter manner and does not focus on website articles as much as television. It is the opposite, and the media had a constant focus on website stories by increasing their average length, complexity, and lexicon richness. Moreover, the thesis results confirmed that CNN presents less negative news, and FOX has a lower negative news ratio than on average when a subject is related to Republican President Donald J. Trump.

Results suggest that CNN and Fox News are highly polarized in article news and television shows. Moreover, a widening drift between media is noticed in articles news, especially after Mueller's report. Results also support the hypothesis that there is a significant gap between negative and neutral news in both media which indicates that sentiment should be taken into account in polarization analysis.

In general, television polarization indicates constant polarization between media and the tendency for anchors to cluster close together. Analysis of the most popular television shows on CNN and FOX anchors indicated that only Brat Baier truly is the down-the-middle, straight-news anchor who only recently started his shift to the far-right. Besides, even though CNN anchors Anderson Cooper, Chris Cuomo, and Don Lemon states being apolitical and unbiased results says otherwise, they are shifted to the left.

The discussed model suggests how Poisson scaling models could incorporate anchors' status and his television shows' popularity indicators such as text sentiment, show viewership data, and more. This model is only suggestive since the results can be validated only by experts in this field who could evaluate television shows' stance in general polarization.

The thesis contributes to the text mining research area by introducing the specific data preprocessing approach for the Poisson scaling models, which follows the stability of polarization's position mean and variance. The process removes not polarized periods and lexicon shifts which are white noise. The comparison results between the function and human manual cleanse work differ insignificantly. An algorithm is constructed to measure leading television figures' influence that incorporates anchors' status and their television shows' popularity indicators.

In further research, graph theory could be introduced to evaluate anchors' coverage of topics – how many leading events, places, and politicians are mentioned. Also, it would allow measuring estimates such as closeness and distances between anchors, giving more insights into the anchors' information to the public and what key terms are used, how connected one presenter is to the other. Furthermore, to enhance the method to prepare data for the model, different theories can be applied, for instance, statistical quality control theory, which has more foundation to work on. Lastly, with the help of a political scientist, presented models could be evaluated and confirmed.

References

[1] A. Abramowitz, J. McCoy, *United States: Racial Resentment, Negative Partisanship, and Polarization in Trump's America, The ANNALS of the American Academy of Political and Social Science,* 2019, **681**(1), p.p. 137–156. https://doi.org/10.1177/0002716218811309

[2] A. I. Abramowitz, S. W. Webster, *Negative Partisanship: Why Americans Dislike Parties But Behave Like Rabid Partisans,* 2018, *Supplement: Advances in Political Psychology,*, **39**(1), p.p. 119-135. https://onlinelibrary.wiley.com/doi/full/10.1111/pops.12479

[3] BBC, 2014, accessed 5 October 2021, https://www.bbc.com/future/article/20140728-why-is-all-the-news-bad

[4] K. Benoit et al., *quanteda: An R package for the quantitative analysis of textual data, Journal of Open Source Software,* 2018, **3**(30), p.p. 137–156. https://doi.org/10.21105/joss.00774

[5] V. Bochkarev, A. Shevlyakova, V. Solovyev, *Average word length dynamics as indicator of cultural changes in society,* 2012, *Social Evolution and History,*, **14**, p.p. 153-175.

[6] Business Insiders, 2019, accessed 7 October 2021, https://www.businessinsider.com/fox-news-cnn-change-evolution-2010-2019-11

[7] A. Casero-Ripolles, *"Impact of Covid-19 on the media system. Communicative and democratic consequences of news consumption during the outbreak, El profesional de la informacio´*, 2020, **29**(2). lhttps://doi.org/10.3145/epi.2020.mar.2

[8] J. Chen, A. K. Gupta, *Parametric Statistical Change Point Analysis,* 2000, p.p. 215-255.

[9] B. Efron, *Bayesian inference and the parametric bootstrap*, *Ann Appl Stat*, 2012, **6**(4), p.p. 1971-1997. doi:10.1214/12-AOAS571

[10] I. Feinerer, K. Hornik, D. Meyer, *Text Mining Infrastructure in R, Journal of Statistical Software,*, 2008, **25**(5). https://doi.org/10.18637/jss.v025.i05

[11] L. Feldman, et al., *Climate on Cable: The Nature and Impact of Global Warming Coverage on Fox News, CNN, and MSNBC,* 2012, *The International Journal of Press/Politics,*, **17**(1), p.p. 3–31. https://doi.org/10.1177/1940161211425410

[12] Forbes, 2021, accessed 1 October 2021, https://www.forbes.com/sites/markjoyella/2021/08/03/fox-news-sweeps-weeks-top-five-most-watched-shows-in-cable-news/?sh=1621f7ee4cd6

[13] Forbes, 2021, accessed 20 December 2021, https://www.forbes.com/sites/markjoyella/2021/03/23/tucker-carlson-leads-fox-news-to-a-big-win-in-weekly-cable-news-ratings/?sh=149f25bb4a1c

[14] J. Grimmer, B. M.Stewart,*Text as data: The promise and pitfalls of automatic content analysis methods for political texts*, *Political Analysis*, 2013, **21**, p.p. 267-297. 10.1007/978-3-642-21551-3-6

[15] P. S. Hart, S. Chinn, S. Soroka,*Politicization and Polarization in COVID-19 News Coverage*, *Science Communication*, 2020, **42**(5). 10.1007/978-3-642-21551-3-6

[16] HuggingFace, accessed September 2 2021. https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment

[17] J. Kaufman, *google-10000-english,* 2021. https://github.com/first20hours/google-10000-english#readme

[18] D. Kouzis-Loukas, 2016, *Learning Scrapy, Packt Publishing Ltd.*

[19] B, E. Lauderdale, A. Herzog, *Measuring Political Positions from Legislative Speech, Political Analysis*, 2016, **24**(3), p.p. 374-394. doi:10.1093/pan/mpw017

[20] M. Laver, K. Benoit, and J. Garry, *Extracting Policy Positions From Political Texts Using Words as Data, American Political Science Review*, 2003, **97**(2): p.p. 311-331.

[21] Los Angeles Times, 2019, accessed 3 October 2021, https://www.latimes.com/business/hollywood/la-fi-ct-suzanne-scott-fox-news-20190403-story.html

[22] J. G. Martin, A. Yurukoglu, *Bias in Cable News: Persuasion and Polarization,* 2017, *American Economic Review,*, **107**(9), p.p. 2565-99. https://www.aeaweb.org/articles?id=10.1257/aer.20160812

[23] Media Matters, E. Hananoki, 2012, accessed 17 September 2021, https://www.mediamatters.org/fox-news/report-30-fox-news-hosts-and-contributors-who-are-campaigning-republicans

[24] S.K. Ng T. Krishnan G. J. Mclachlan, *The EM algorithm, In: J. Gentle, W. Härdle, Y. Mori (eds) Handbook of Computational Statistics. Springer Handbooks of Computational Statistics. Springer, Berlin, Heidelberg*,2004. https://doi.org/10.1007/978-3-642-21551-3_6

[25] J. M. Pérez, J. C. Giudici, F. Luque, *Pysentimiento: A Python Toolkit for Sentiment Analysis and SocialNLP tasks,* 2021. https://arxiv.org/abs/2106.09462

[26] Pew Research Center, *U.S. Media Polarization and the 2020 Election: A Nation Divided* 2020.

[27] Pew Research Center, 2020, accessed 1 September 22 2021, https://www.pewresearch.org/fact-tank/2020/04/01/americans-main-sources-for-political-news-vary-by-party-and-age/

[28] Pew Research Center, *In Changing News Landscape, Even Television is Vulnerable*, 2012. https://www.pewresearch.org/politics/2012/09/27/section-4-demographics-and-political-views-of-news-audiences/

[29] P. Rozin E. B. Royzman, *Negativity Bias, Negativity Dominance, and Contagion,* 2001, *Personality and Social Psychology Review,*, **5**(4), p.p. 296–320. https://doi.org/10.1207/S15327957PSPR0504_2

[30] A. Schofield, D. Mimno, *Comparing Apples to Apple: The Effects of Stemmers on Topic Models, Transactions of the Association for Computational Linguistics,* 2016, **4**, p.p. 287-300. https://doi.org/10.1162/tacl_a_00099

[31] S. S. Shivaji, 2014, *Selenium Webdriver in Python: Learn with Examples,*, *CreateSpace Independent Publishing Platform.*

[32] J. B. Slapin, S. O. Proksch, *A Scaling Model for Estimating Time-Series Party Positions from Texts, American Journal of Political Science*, 2008, **52**(3): p.p. 52, 705-772. doi.org/10.1111/j.1540-5907.2008.00338.x

[33] The Sun, 2021, accessed 15 December 2021, https://www.the-sun.com/news/2131092/donald-trump-impeached-the-first-time-congress/

[34] The Hollywood Reporter, 2018, accessed 1 December 2021, https://www.hollywoodreporter.com/tv/tv-news/who-are-americas-trusted-tv-news-anchors-poll-1160597/

[35] D. Thomas, *Type-token Ratios in One Teacher's Classroom Talk: An Investigation of Lexical Complexity,* 2005.

[36] H. Wickham, *Stringr: Modern, consistent string processing, The R Journal,* 2010, **2**(2). https://doi.org/10.32614/RJ-2010-012

[37] Yinhan Liu, et al., *RoBERTa: A Robustly Optimized BERT Pretraining Approach,* 2019. https://arxiv.org/abs/1907.11692

Appendices

A Polarization graphs

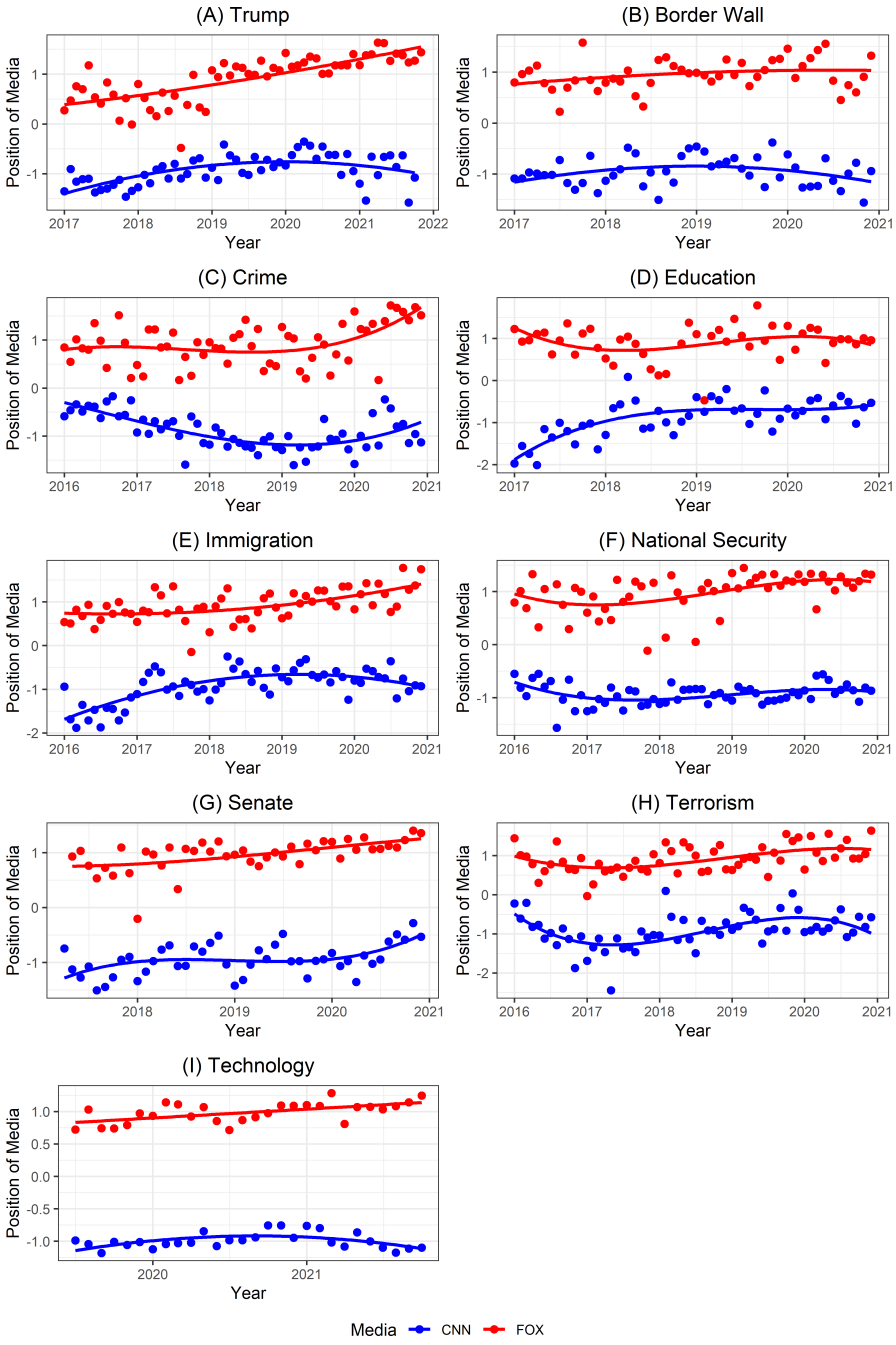

Figure 19: Polarization by *Wordfish* with coefficient p chosen manually for every topic

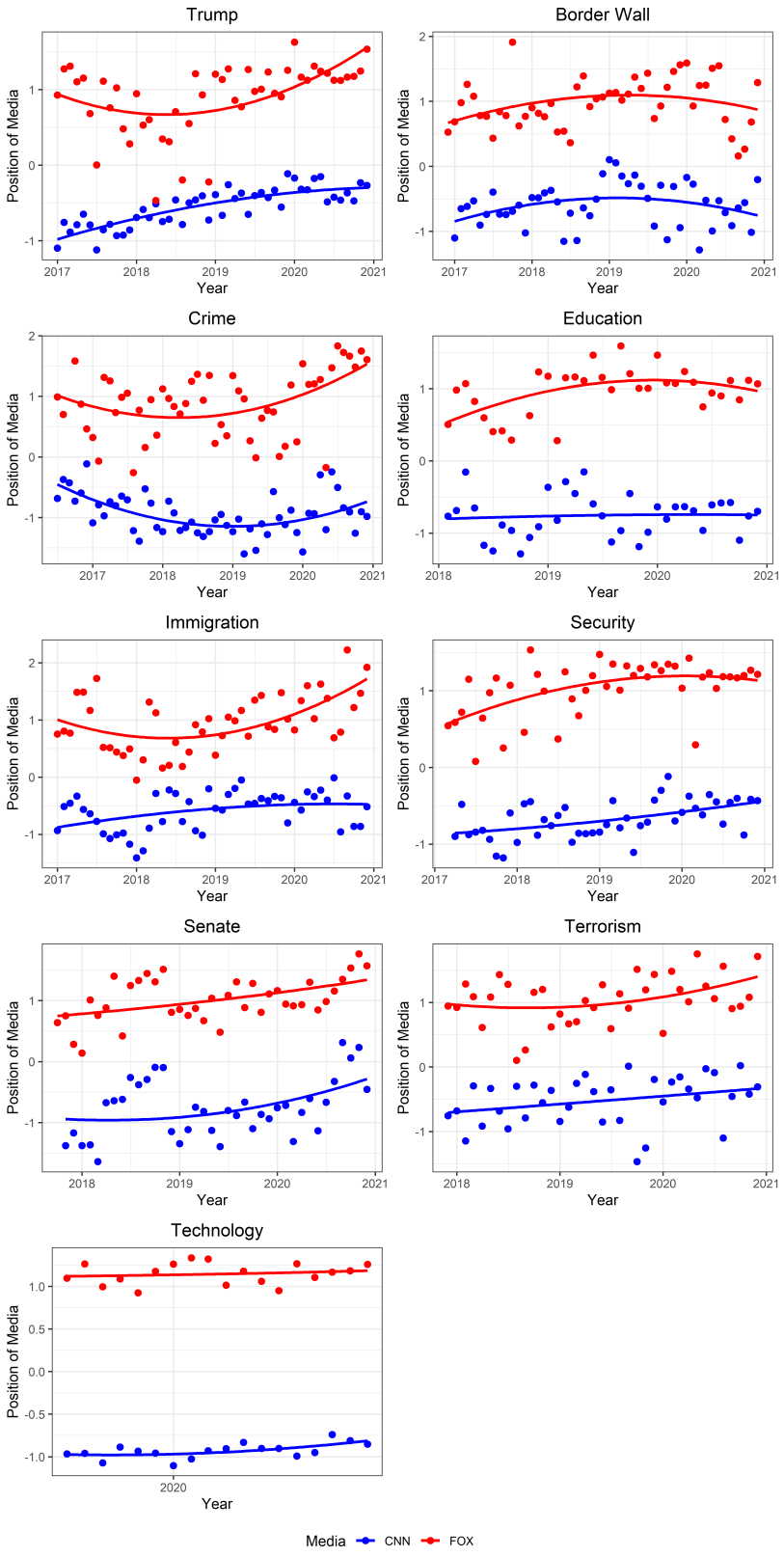

Figure 20: Polarization by *Wordfish* with coefficient p suggested by the method (after step 2) for every topic

B Scraping FOX news website

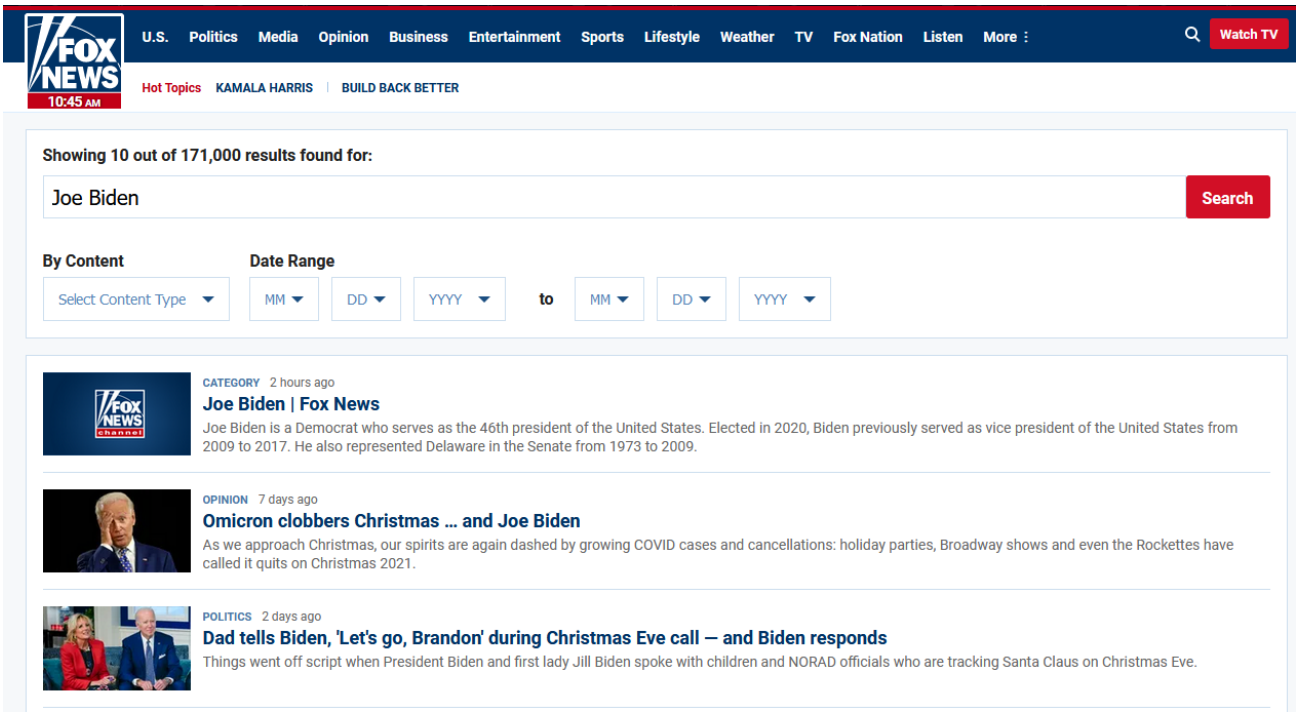

Figure 21: FOX news websites search page layout

Fox News website puts a shortlist of 100 articles per search; hence, it cannot be scraped without interference. A researcher has to follow the steps to scrape text from dynamical page (see Figure 21):

1. Select the "By Content" button and click "Articles" to get only articles.
2. Locate and select the "Date Range" buttons to obtain specified date period articles. This step has to be set manually because some subjects get more attention sometimes. For instance, "Government Shutdown" had approximately ten times more articles around the beginning of 2019 than casually.
3. Press the "Search" button and when the page loads up, scroll down and click the "Show More" button nine times.
4. Parse every article link and then scrape text from it.

```
import scrapy
import time
import random
from scrapy.selector import Selector
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
```

```
from selenium.common.exceptions import TimeoutException
from selenium.webdriver.support.ui import Select

class FoxSpider(scrapy.Spider):
    #  Spiders name to be called in terminal
    name = 'fox'

    allowed_domains = ['www.foxnews.com']
    start_urls = ['https://www.foxnews.com']

    def __init__(self):
        chrome_options = Options()
        driver = webdriver.Chrome(executable_path=str('./chromedriver')
                                  , options=chrome_options)

        #  Determine an initial page, and keyword to look after
        driver.get("https://www.foxnews.com/search-results/search?q=Joe%20Biden")

        #  Find "By Content" button and select "Articles"
        driver.find_element_by_xpath(
            '//div[@class = "filter content"]//button[@class = "select"]'
            ).click()

        driver.find_element_by_xpath(
            '//div[@class = "filter content"]//label//input[@title="Article"]'
            ).click()

        #  Find "Date Range" from--to fields
        #  Start month
        driver.find_element_by_xpath(
            '//div[@class = "date min"]//div[@class = "sub month"]//button[@class="
                select"]'
            ).click()
        driver.find_element_by_id("01").click()

        #  Start day
        driver.find_element_by_xpath(
            '//div[@class = "date min"]//div[@class = "sub day"]//button[@class="
                select"]'
            ).click()
        driver.find_element_by_xpath(
            '//div[@class = "date min"]//div[@class = "sub day"]//ul[@class="option
                "]//li[@class="01"]'
            ).click()

        #  Start year
        driver.find_element_by_xpath(
            '//div[@class = "date min"]//div[@class = "sub year"]//button[@class="
                select"]'
            ).click()
        driver.find_element_by_id("2016").click()
```

```
57
58          #  End month
59          driver.find_element_by_xpath(
60              '//div[@class = "date max"]//div[@class = "sub month"]//button[@class="
                    select"]'
61              ).click()
62          driver.find_element_by_xpath(
63              '//div[@class = "date max"]//div[@class = "sub month"]//ul[@class="option
                    "]//li[@class="01"]'
64              ).click()
65
66          #  End day
67          driver.find_element_by_xpath(
68              '//div[@class = "date max"]//div[@class = "sub day"]//button[@class="
                    select"]'
69          ).click()
70          driver.find_element_by_xpath(
71              '//div[@class = "date max"]//div[@class = "sub day"]//ul[@class="option
                    "]//li[@class="01"]'
72              ).click()
73
74          #  End year
75          driver.find_element_by_xpath(
76              '//div[@class = "date max"]//div[@class = "sub year"]//button[@class="
                    select"]'
77          ).click()
78          driver.find_element_by_xpath(
79              '//div[@class = "date max"]//div[@class = "sub year"]//ul[@class="option
                    "]//li[@id="2017"]'
80              ).click()
81
82          #  Click "Search" icon
83          select = driver.find_element_by_xpath('//div[@class = "button"]')
84          select.click()
85
86          #  Define wait time to act more like human and not to put on load on the page
87          wait = WebDriverWait(driver, 10)
88
89          #  Page ojnly shows 100 articles per search, hence only 9 "Show more" buttons
90          i = 0
91          while i < 9:
92              try:
93                  time.sleep(3)
94
95                  #  Wait until page loads the button and then press it, if no BREAK
96                  element = wait.until(EC.visibility_of_element_located(
97                      (By.XPATH, "(//div[@class='button load-more'])[1]/a")))
98                  element.click()
99                  i += 1
100             except TimeoutException:
101                 break
```

```
            #  To not repeat actions in the same manners generate different waits
            time.sleep(random.randint(2, 7))

        #  Copy what is shown in page after all "Show more" buttons are pressed
        self.html = driver.page_source

    #  Parser, which gets separate articles data from the main source
    def parse(self, response):
        resp = Selector(text=self.html)

        #  Read every title in the html saved
        results = resp.xpath("//article[@class='article']//h2[@class='title']/a")

        for result in results:
            title = result.xpath(".//text()").get()
            link = result.xpath(".//@href").get()

            #  Read link which have to be scraped and send its title as metadate
            yield response.follow(url=link, callback=self.parse_article, meta={"title
                ": title})

    def parse_article(self, response):
        title = response.request.meta['title']
        authors = response.xpath("(//div[@class='author-byline']//span/a)[1]/text()")
            .getall()

        #  If there is no author look for higher level entity
        if len(authors) == 0:
            authors = [i for i in response.xpath("//div[@class='author-byline opinion
                ']//span/a/text()").getall() if 'Fox News' not in i]

        #  Text is mostly separated by paragraphs, it must be joined
        content = ' '.join(response.xpath("//div[@class='article-body']//text()").
            getall())

        yield {
            "title": title,
            "author": ' '.join(authors),
            "time": response.xpath("//div[@class='article-date']/time/text()").get(),
            "content": content
        }
```

Listing 1: Python script to extract article text from FOX website using Selenium and Scrapy packages

C Scraping CNN news website

CNN website (see Figure 22) differs by margin, nonetheless it requires modified scraping steps:

1. Select the "News" choice option button and click on the "Stories" button to get only articles.

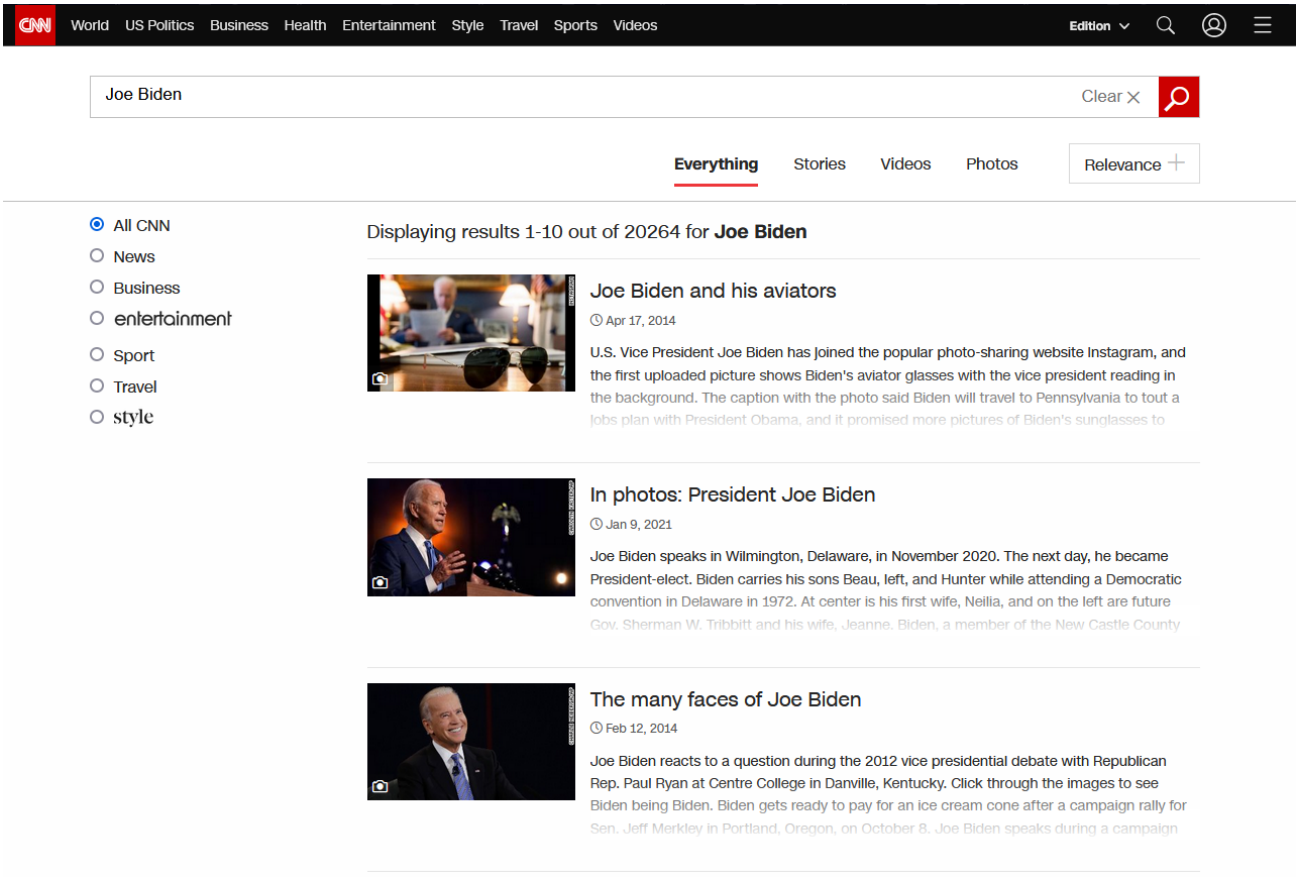

Figure 22: CNN news websites search page layout

2. Locate and select the "Relevance" buttons, which outputs only the most important and visited articles with that keyword. This has to be done because of an error in the CNN website, and the site shows how many articles they have by keyword. However, only can 10000 stories can be scraped at most.
3. Press the "Search" button and when the page loads up, scroll down, save HTML code and click the "Next Page" button. Repeat it 999 times.
4. Parse every article link and then scrape text from it.

```
import scrapy
import time
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
from scrapy.selector import Selector

class CnnSpider(scrapy.Spider):
    #  Spiders name to be called in terminal
    name = 'cnn'

    allowed_domains = ['edition.cnn.com']
    start_urls = ['https://edition.cnn.com']
```

```
16
17      def __init__(self):
18          chrome_options = Options()
19          driver = webdriver.Chrome(executable_path=str('./chromedriver')
20              , options=chrome_options)
21
22          #  Link should be copied from the website, it is complex and error can accur
23          driver.get("https://edition.cnn.com/search?q=Donald%20Trump&size=10&category=
                us,politics,world,opinion,health&type=article&sort=relevance")
24
25          #  Create a wait function to imitate human-like behavour
26          wait = WebDriverWait(driver, 10)
27
28          #  Copy the first page html code
29          self.html = [driver.page_source]
30
31          # start turning pages (there will not be more than 1000 because of the error)
32          i = 0
33          while i < 1000:
34              i += 1
35
36              #  Button do not load up that fast
37              next_btn = wait.until(EC.visibility_of_element_located(
38                      (By.XPATH
39                      , "(//div[@class='pagination-arrow pagination-arrow-right
                            cnnSearchPageLink text-active'])")))
40              next_btn.click()
41
42              #  Append a new page html source to already obtained
43              self.html.append(driver.page_source) # not the best way but will do
44
45      def parse(self, response):
46          #  Loop though pages in saved html list
47          for page in self.html:
48
49              #  Convert html to text
50              resp = Selector(text=page)
51
52              #  Put articles to a list for another loop
53              results = resp.xpath("//div[@class='cnn-search__result cnn-search__result
                    --article']/div/h3/a")
54
55              for result in results:
56                  title = result.xpath(".//text()").get()
57
58                  # Avoid videos, COVID related news or ads
59                  if ("Video" in title) | ("coronavirus" in title) | ("http" in title):
60                      continue
61                  else:
62                      #  Remove domain in the beginning of a link
63                      link = result.xpath(".//@href").get()[13:]
```

```
64
65                      #  Determine a link you want to scrape and send title as metadata
66                      yield response.follow(url=link
67                          , callback=self.parse_article
68                          , meta={"title": title})
69
70      # pass on the links to open and process actual news articles
71      def parse_article(self, response):
72          title = response.request.meta['title']
73
74          # several variations of author's locator
75          authors = response.xpath(
76              "//span[@class='metadata__byline__author']//text()"
77              ).getall()
78
79          #  In case an article do not have an author find a higher entity(subtitle)
80          if len(authors) == 0:
81              authors = response.xpath(
82                  "//p[@data-type='byline-area']//text()"
83                  ).getall()
84              if len(authors) == 0:
85                  authors = response.xpath(
86                      "//div[@class='Article__subtitle']//text()"
87                      ).getall()
88
89          #  Connect separated texts and if it is None convert it to a blank line
90          content = ' '.join(response.xpath(
91              "//section[@id='body-text']/div[@class='l-container']//text()")
92              .getall())
93
94          if content is None:
95              content = ' '
96
97          yield {
98              "title": title,
99              "author": ' '.join(authors), # could be multiple authors
100             "time": response.xpath("//p[@class='update-time']/text()").get(),
101             "content": content
102         }
```

Listing 2: Python script to extract article text from CNN website using Selenium and Scrapy packages

D Text data preparation to textual analysis models

Function to clean FOX media textual data, as is described in section 2.2.3:

```
1  html.remainders = c() #  Include all html terminology by your experience, for
       instance, c(containerid", "typeof"...)
2
3  preprocess_fox <- function(fox_dataset
4                             , remove_words #  Words you want to remove from title and
                                 content
```

```
                          , output_name #  Output location and name to save csv
) {
#  Encoding to avoid random symbols in a text
for (col in colnames(fox_dataset)){
Encoding(fox_dataset[[col]]) <- "UTF-8"}

#  Words to avoid for a topic
if (!is.null(NULL)) {
  fox_dataset = fox_dataset[!grepl(remove_words, fox_dataset$title),]
}

#  Time column, align timestamps
fox_dataset[((str_sub(fox_dataset$time,-4,-3) != '20') & (str_sub(fox_dataset$time
    ,3,6) != 'days')), "time"] <- paste0(fox_dataset[((str_sub(fox_dataset$time
    ,-4,-3) != '20') & (str_sub(fox_dataset$time,3,6) != 'days')), "time"], ", 2021"
    )

#  Remove whitespaces, change a format
fox_dataset$time = str_squish(fox_dataset$time)
fox_dataset$time <- as.Date(fox_dataset$time,format='%B %d, %Y')
fox_dataset = na.omit(fox_dataset)

#  Delete symbols, if symbol is considered to be significant, change it to a word
fox_dataset$author <- sapply(fox_dataset$author, function(x) gsub("[-|=|\\+]", "",
     x))
fox_dataset$title <- sapply(fox_dataset$title, function(x) gsub("\\$", "dollar", x)
    )
fox_dataset$content <- sapply(fox_dataset$content, function(x) gsub("\\$", "dollar"
    , x))
fox_dataset$author = trimws(fox_dataset$author)

#  Title column
documents = fox_dataset[,'title']
documents <- Corpus(VectorSource(documents))

funs <- list(content_transformer(tolower)
             , removeNumbers
             , removePunctuation)
documents = tm_map(documents, FUN = tm_reduce, tmFuns = funs)[[1]]

documents = tm_map(documents, removeWords, c(stopwords("english"), "close", "video"
    ))
fox_dataset[,'title'] = documents

#  Clean Author column, only a first author is considered
fox_dataset[,'author'] = sub(",.*", "", fox_dataset$author)
fox_dataset[,'author'] = sub(" and.*", "", fox_dataset$author)

#  Clean author column
documents = fox_dataset[,'author']
documents <- Corpus(VectorSource(documents))
```

```
  documents = tm_map(documents, removeNumbers)
  documents = tm_map(documents, removeWords, c("rep"))
  fox_dataset[,'author'] = documents


  #  Content column
  documents = fox_dataset[,'content']
  documents <- Corpus(VectorSource(documents))
  documents = tm_map(documents, content_transformer(tolower))
  documents = tm_map(documents, removeNumbers)

  #  After stop words are cleaned, HTML remainders have to be removed because they
      shift the model output
  documents = tm_map(documents, removeWords, c(stopwords("english"), "close", "video"
      , html.remainders[]))
  documents = tm_map(documents, removePunctuation,ucp = TRUE)
  documents = lapply(documents, function(x) gsub("[^\\s]*>[^\\s]*", "",x))

  #  Words up to 2 letters and longer than 15 have to be removed, they dont give a
      lot of information
  #  and occur rarely
  documents = lapply(documents, function(x) rm_nchar_words(x, "1,2"))
  documents = lapply(documents, function(x) rm_nchar_words(x, "15,"))

  #  Separate original content and stemmed, in case both datasets could be used
  fox_dataset$content.original = unlist(documents,use.names = FALSE)

  documents <- Corpus(VectorSource(documents))
  documents <- tm_map(documents, stemDocument, language = "english")
  fox_dataset[,'content'] = documents

  #  Rename output columns and save it to a excel sheet
  colnames(fox_dataset) = c("title", "author", "time", "content_stem", "content_
      original")
  write.csv(fox_dataset, paste0(output_name,"_fox_cleaned.csv"), row.names=FALSE,
      quote=FALSE)
}
```

Listing 3: R script to clean scraped text

E Data chunking and Roberta model application for a sentiment ratio extraction

Television shows sentiment distribution has been evaluated by following steps:

1. The "cardiffnlp/twitter-roberta-base-sentiment" model for sentiment analysis has been chosen from the "huggingface" framework since it is fine-tuned on Twitter tweets no longer than 150 words which is suchlike a paragraph in a text. More information in [16].
2. The text was tokenized and separated into smaller chunks of 100 tokens to present data similarly to a tweet format. Next, an attention mask array for every chunk has been generated and stacked together with input ids to create a dictionary that fulfills the model requirements.

3. The model results are logits that have to be transformed to probabilities, then ranked from lowest to the highest, and then used to calculate how many paragraphs in a show have a negative or neutral sentiment.

```
!pip install transformers

import torch
import numpy as np
import pandas as pd
from scipy.special import softmax
from transformers import AutoTokenizer, AutoModelForSequenceClassification

MODEL = f"cardiffnlp/twitter-roberta-base-sentiment"

tokenizer = AutoTokenizer.from_pretrained(MODEL)
model = AutoModelForSequenceClassification.from_pretrained(MODEL)
model.save_pretrained(MODEL)

chunksize = 100

def chunk_text(txt):
  #  Tokenize text
  tokens = tokenizer.encode_plus(txt, add_special_tokens=False, return_tensors='pt')

  #  Split text to chunks of 100 tokens
  input_id_chunks = tokens['input_ids'][0].split(chunksize-2)
  mask_chunks = tokens['attention_mask'][0].split(chunksize-2)

  input_id_chunks = list(input_id_chunks)
  mask_chunks = list(mask_chunks)

  #  Add the beginning, ending tokens to chunked arrays;
  for i in range(len(input_id_chunks)):
    input_id_chunks[i] = torch.cat([
        torch.Tensor([101]), input_id_chunks[i], torch.Tensor([102])
    ])
    mask_chunks[i] = torch.cat([
        torch.Tensor([1]), mask_chunks[i], torch.Tensor([1])
    ])

    pad_len = chunksize - input_id_chunks[i].shape[0]
    if pad_len > 0:
      input_id_chunks[i] = torch.cat([
        input_id_chunks[i], torch.Tensor([0] * pad_len)
      ])
      mask_chunks[i] = torch.cat([
        mask_chunks[i], torch.Tensor([0] * pad_len)
      ])

  #  Stack chunks to one tensor
  input_ids = torch.stack(input_id_chunks)
  attention_mask = torch.stack(mask_chunks)
```

```
49
50    #  Create a map for the model later
51    input_dict = {
52        'attention_mask' : attention_mask.int(),
53        'input_ids' : input_ids.long()
54    }
55
56    return input_dict
57
58  #  Dataframe to store negative/neutral sentiment columns
59  df = pd.DataFrame(columns = ['Negative', 'Neutral'])
60
61  #  Read a dataframe
62  data = pd.read_csv('anchor_shows.csv')
63
64  for i in range(len(data)):
65
66    #  Chunk text with the function
67    input_dict = chunk_text(data[i,'content'])
68
69    #  Feed it to the model
70    output = model(**input_dict)
71
72    #  Calculate shows' perentage of negative/neutral paragraphs
73    #  Transform data to see which sentiment it mostly is
74    scores = output[0].detach().numpy()
75    scores = softmax(scores)
76    ranking = np.argsort(scores)
77    ranking = ranking[::-1]
78
79    #  Calculate percentage of negative and neutral paragraphs
80    sent_0 = np.count_nonzero(ranking[:,2] == 0)/len(ranking[:,2])
81    sent_1 = np.count_nonzero(ranking[:,2] == 1)/len(ranking[:,2])
82
83    #  Fit it back into a dataframe
84    df = df.append({'Negative' : sent_0, 'Neutral' : sent_1}, ignore_index=True)
85
86  #  Concat existing dataframe with sentiment columns
87  result = pd.concat([data, df], axis=1)
88
89  #  Write it back to other csv
90  result.to_csv('anchor_shows_sentiment_added.csv')
```

Listing 4: Python script to chunk long closed captions of a show and evaluate average sentiment of it using deep learning method Roberta

F Function to remove lexicon shift and redundant terms

```
1  library(data.table)
2  library(stringr)
```

```
3  library(tm)
4  library(changepoint)
5
6
7  #  Function takes data and p coefficient as input and returns wordfish estimates
8  run.wordfish = function(x  #  dataframe
9                          , p  #  p coefficient to remove noise
10 ) {
11   #  Find number of documents which is used later to removed rare terms
12   ndocs = length(x)
13
14   #  Remove every term that is not in at least ndocs*p documents
15   dtm = DocumentTermMatrix(x, control = list(bounds = list(global = c(ndocs * p,
         ndocs))))
16
17   #  Convert it to the dfm format
18   dfm <- quanteda::as.dfm(dtm)
19
20   #  Trim terms that appear less than 5 times in a document, mostly it is mistype
         trimming
21   dfm <- quanteda::dfm_trim(dfm, min_docfreq = 5)
22
23   #  Run the model
24   mywf <- textmodel_wordfish(dfm, dir = c(1,2))
25
26   #  Create a dataframe to test on
27   df.wf = cbind(summary(mywf)[[2]], x[,c("Year", "Media")])
28
29   return(df.wf)
30 }
31
32 #  Remove noise and check outliers positioning
33 remove.noise = function(x  #  dataframe
34       , p  #  p coefficient to remove noise
35       , o  #  threshold for outliers
36       , i  #  p increasement step size
37 ) {
38   #  Run wordfish
39   df.wf = run.wordfish(x,p)
40
41   #  Find all different medias
42   sides = df.wf$Media %>% unique()
43
44   #  Separate sides
45   df.wf_1 = df.wf %>% filter(Media == sides[1])
46   df.wf_2 = df.wf %>% filter(Media == sides[2])
47
48   #  Measure how many values are far away from the trend (in opposite side)
49   outliers1 = sum(df.wf_1$theta>0)/length(df.wf_1$theta)
50   outliers2 = sum(df.wf_2$theta<0)/length(df.wf_2$theta)
51
```

```
  #  If more than o% of values are outliers then increase coefficient p and repeat
  if (outliers1 > o | outliers2 > o) {
    if (p == 1) {
    } else {
      p = p + i
    }
  }

  return(p)
}


#  Function removes change points after remove.noise function converge p
remove.changepoints = function(df.wf  #  dataframe   of wordfish results
    , m  #  mean threshold to remove only significantly shifted mean periods
) {
  #  Create vectors for further analyzed data to store
  changepoint.df = NULL
  changepoint.means = NULL

  #  Find names for separetion of media
  sides = df.wf$Media %>% unique()

  #  Find changepoint measures
  for (media in sides) {
    #  Separate for every media
    df_x = df.wf %>% filter(Media == media)

    #  Apply changepoint package to obtain possible shifts in mean or variance
    fit_changepoint = cpt.meanvar(df_x$theta, minseglen=4)

    #  Store it
    changepoint.df = c(changepoint.df, cpts(fit_changepoint))

    #  Take out mean measures of possible shifter intervals
    fit.means = param.est(fit_changepoint)$mean

    #  Measure a shift to later compare it with the threshold
    changepoint.means = c(changepoint.means, fit.means[2] - fit.means[1])
  }

  #  Find the shortest shift to determine if both sides moved significantly
  chp.min = min(changepoint.df)

  #  If one of media didnt have a change-point NA can ruin the flow
  changepoint.means = na.omit(changepoint.means)[1]

  #  If the shift is bigger that m value then delete that shift in both sides
  if (max(abs(changepoint.means)) > m & is.null(changepoint.means) == F) {
    #  remove top values of shifter period, multiplied by to to remove both media
        data
```

```
    df.wf = tail(df.wf, -chp.min*2)
  }

  #  Find the lowest date in analyzed period to know what values to filter in the
      next step
  min.date = min(df.wf$Year)
  return (min.date)
}



#  In step 3 variance is considered to optimize the coefficient p
find.variance = function(df.wf   #  dataframe
                         , m  #  Coefficient p
) {
  #  Run wordfish
  df.wf = run.wordfish(x,p)

  #  Detect date when a shift happens
  min.date = remove.changepoints(df.wf,m)

  #  Filter everything up to that date
  df.wf = df.wf %>% filter(Year > min.date)
  #  Find all different medias
  sides = mywf$Media %>% unique()

  #  Separate sides
  df_1 = df %>% filter(Media == sides[1])
  df_2 = df %>% filter(Media == sides[2])

  var1 = var(df_1)
  var2 = var(df_2)

  return(c(var1,var2))
}

#  The algorithm
the.method = function(x) {

  #  Initial parameters
  p = 0.4
  o = 0.05
  i = 0.02
  m = 0.4
  v = 0.01

  #  Step 1, remove the general noise by detecting number of outliers
  while (p_change > 0) {
    p_new = remove.noise(x  #  Dateframe
                         , p #  Begin with initial p
                         , o  #  threshold for outliers
```

```
                          , i  #  p increasement step size
    )

    #  Check if p increased
    p_change = p_new - p

    #  Set initial p to the new p value
    p = p_new
  }

  #  Step 2 find change points for the output
  df.wf = run.wordfish(x,p)
  df.wf_next = run.wordfish(x,p+i)
  min.date = remove.changepoints(df.wf, m)

  #  find.variance function has a already in-built function to remove
  #  change points, hence we can care about stability of variance only
  var.change.vector = find.variance(df.wf,m) - find.variance(df.wf_next,m)

  #  Run the flow until variance do not stabilizy by more than 0.1
  while (var.change.vector[1] > v & var.change.vector[2] > v) {
    #  Increace p
    p = p + i

    #  To find new variance change
    df.wf = run.wordfish(x,p)
    df.wf_next = run.wordfish(x,p+i)

    #  Recalculate the variance
    var.change.vector = find.variance(df.wf,m) - find.variance(df.wf_next,m)
  }

  #  Run linear regression to give a notice to a user if there is still a trend
  model = lm(theta~Year, data=df.wf)

  if (coef(model)[2] > r) {
    print("Topic should be revised and put into subtopics to remove additional noise"
        )
  }

  #  Return p value and min.date to filter periods with shifted periods
  return(c(p,min.date))
}
```

Listing 5: R script to evaluate coefficent p