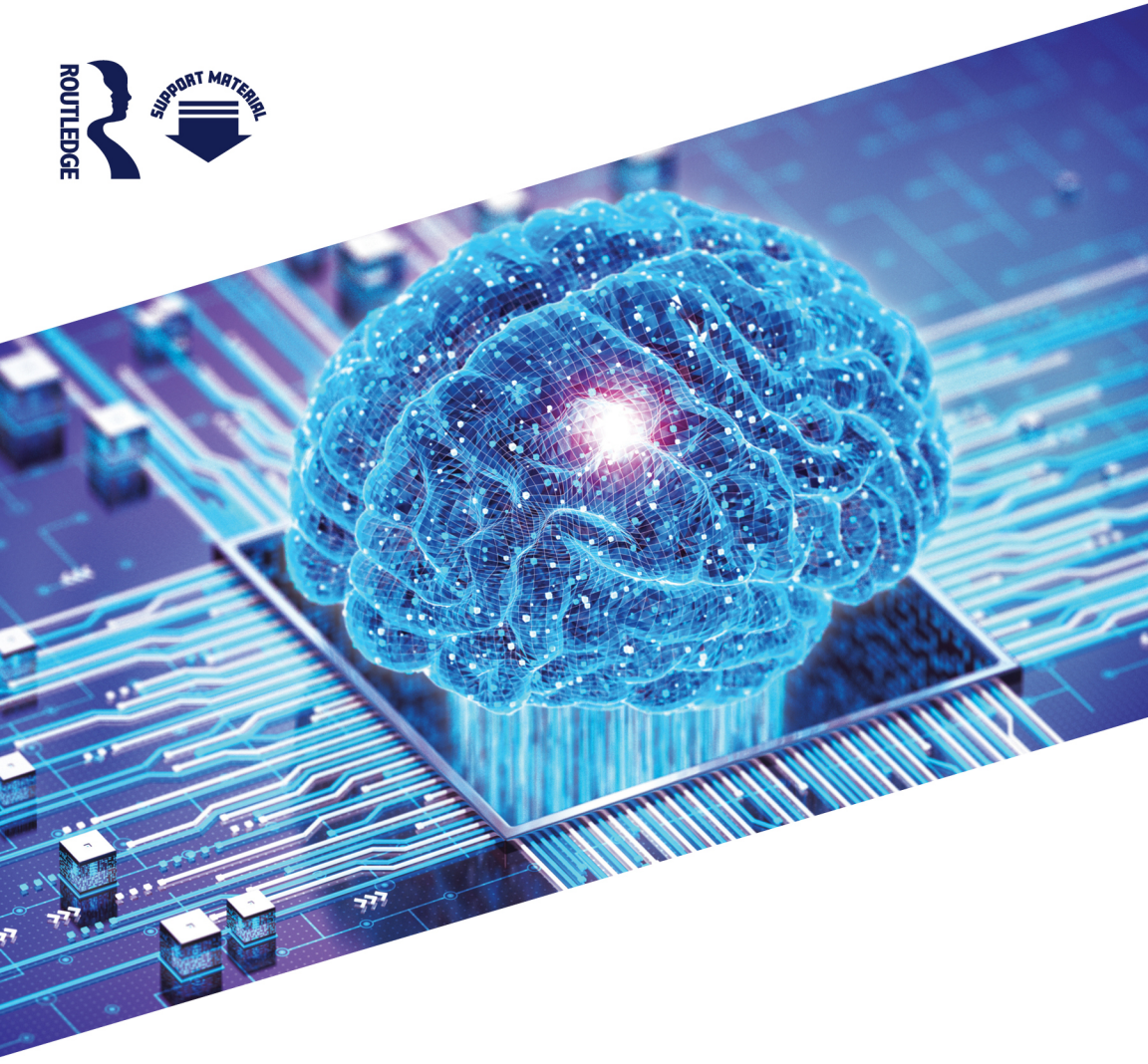
ROUTLEDGE

SUPPORT MATERIAL



THE COMPUTATIONAL CONTENT ANALYST

Using Machine Learning to Classify Media Messages

Chris J. Vargo

The Computational Content Analyst

Most digital content, whether it be thousands of news articles or millions of social media posts, is too large for the naked eye alone. Often, the advent of immense datasets requires a more productive approach to labeling media beyond a team of researchers. This book offers practical guidance and Python code to traverse the vast expanses of data—significantly enhancing productivity without compromising scholarly integrity. We'll survey a wide array of computer-based classification approaches, focusing on easy-to-understand methodological explanations and best practices to ensure that your data is being labeled accurately and precisely. By reading this book, you should leave with an understanding of how to select the best computational content analysis methodology to your needs for the data and problem you have.

This guide gives researchers the tools they need to amplify their analytical reach through the integration of content analysis with computational classification approaches, including machine learning and the latest advancements in generative artificial intelligence (AI) and large language models (LLMs). It is particularly useful for academic researchers looking to classify media data and advanced scholars in mass communications research, media studies, digital communication, political communication, and journalism.

Complementing the book are online resources: datasets for practice, Python code scripts, extended exercise solutions, and practice quizzes for students, as well as test banks and essay prompts for instructors. Please visit www.routledge.com/9781032846354.

Chris J. Vargo is an Associate Professor in the College of Media, Communication, and Information and Leeds School of Business (Courtesy) at the University of Colorado Boulder, USA. His research primarily focuses on the intersection of computational media analytics and political communication, employing computational methods to enhance understanding in these areas.

"*The Computational Content Analyst* opens new research frontiers using highly sophisticated computer-based approaches that greatly expand the substantive depth and scope of quantitative content analysis. These approaches vastly improve scholars' ability to examine the large body of content available on the internet."

—**Maxwell McCombs**, *University of Texas at Austin, USA*

"*The Computational Content Analyst* provides a practical and informative guide for scholars and practitioners aiming to learn the basics of computational approaches to analyzing text. This book is practical and insightful; Vargo makes a complex topic accessible through insightful examples and useful research case studies."

—**Matthew Weber**, *Rutgers University, USA*

"This book makes computational content analysis as easy as following a recipe."

—**Milad Minooie**, *Kennesaw State University, USA*

The Computational Content Analyst

Using Machine Learning to Classify Media Messages

Chris J. Vargo

Routledge
Taylor & Francis Group
NEW YORK AND LONDON

Designed cover image: © BlackJack3D/Getty Images

First published 2025
by Routledge
605 Third Avenue, New York, NY 10158

and by Routledge
4 Park Square, Milton Park, Abingdon, Oxon, OX14 4RN

Routledge is an imprint of the Taylor & Francis Group, an informa business

ISBN: 9781032846354 (hbk)
ISBN: 9781032846309 (pbk)
ISBN: 9781003514237 (ebk)

DOI: 10.4324/9781003514237

Typeset in Galliard
by Apex CoVantage, LLC

Access the Support Material: www.routledge.com/9781032846309

To my doctoral advisors and mentors, Dr. Joe Bob Hester and Dr. Jaime Arguello at the University of North Carolina at Chapel Hill. I will be forever grateful for you encouraging me to learn Python and take classes in information science. Until that point, I was not yet truly a nerd. Thank you for letting me join the club.

Contents

Preface

In every corner of our lives, unbeknownst to us, we are constantly communicating in data. There is too much data in our world to make complete sense of but we have developed ways to filter or make sense of large data. In my media analytics courses, one way we talk about doing this is by "segmenting" the data. Whether we segment a list of customers based on their likelihood to buy a product, or simply segment a weekly grocery list by food groups, categories, and dietary needs, we, in our own ways, are constantly organizing and filing away the data that is thrown at us in life.

Inherent in newspapers, movies, artwork, speeches, and even TikTok posts are overt and covert messages with profound implications. Content analysis breathes life into these messages. This research technique makes replicable and valid inferences by interpreting and coding material. Content analysis aids us in deconstructing and discerning the intricacies of media and communication. Through meticulous coding and examination, we can reveal insights into structure, meaning, and patterns. In this book, we'll focus on coding and analyzing various types of media, from social media posts to news articles and even images.

With the escalating growth of media content, conducting manual content analysis becomes a Herculean task. If you've struggled to do a content analysis, this book is for you. We'll leverage machine learning—a subset of artificial intelligence—and train (or, in generative AI use cases, simply use) a computer model to make accurate predictions or decisions without being explicitly programmed to do so.

This book will include all the methodological considerations one should consider when taking a content analysis and attempting to automate, or at least expedite, that process using computers. We'll review different approaches to automated content analysis, from time-tested approaches like keyword lexicons to supervised machine learning, unsupervised machine learning (e.g., topic modeling), and even emergent techniques using large language models and generative artificial intelligence.

I encourage you to imagine the power of content analysis and machine learning combined. This fusion is a burgeoning area of study that has the potential to streamline media research, making it quicker, easier, and more efficient. The goal of this book is to illuminate that sweet spot where these two fields intersect. It is designed to equip scholars at every stage—from early career researchers to experienced academicians—with the tools to harness automated processes for expediting content analysis.

This book is a "roll up your sleeves" kind of guide. It takes a hands-on approach to understanding and applying machine-learning techniques to content analysis. Each chapter is packed with practical examples and Python code, designed to enable you to navigate through real-world datasets.

It is with much pleasure that I present *The Computational Content Analyst: Using Machine Learning to Classify Media Messages*, a navigator for your journey into the convergence of content analysis and machine learning—the next generation of analyzing media messages.

Chapter 1

Unveiling Content Analysis in the Contemporary Media Ecosystem

Before embarking on my academic career, I experienced the mundane world of data entry firsthand during a memorable "bring your child to work day" at the bank where my mother was employed. At the age of 16, even in a state of partial attentiveness, I could discern the pitfalls inherent in the manual processing of data. Missteps such as typographical errors in postal addresses, the challenges of deciphering illegible handwriting, and a discernible lack of diligence from the individual inputting the data (me) inevitably introduced inaccuracies into the spreadsheet I was assembling. Although the list consisted merely of rewards for children who had adeptly accumulated savings, the implications were troubling, especially given the institution's fiduciary responsibilities.

Data annotation, which encompasses labeling, segmenting, and annotating, is an integral part of the development of machine learning and artificial intelligence systems (Russell & Norvig, 2016). Despite the diligence of human annotators, the manual aspect of labeling data introduces the potential for error, affecting the quality of the resulting data sets (Goodfellow et al., 2016). Computers, though typically *reliable* in that they do the same thing over and over following their strict programming, possess limitations in recognizing and categorizing data in *externally valid* ways, which can lead to systematic errors (Jordan & Mitchell, 2015). In the realm of machine learning, labeling raw data such as images, texts, or videos with descriptors provides essential context, equipping algorithms with the data they need to learn and make predictions (LeCun et al., 2015).

To illustrate, a facial recognition system needs each image to be tagged with names so that it can learn to identify individuals correctly (Taigman et al., 2014). However, these algorithms, sophisticated as they may be, are susceptible to inaccuracies. An instance of this can occur when an individual changes their hair color or style, or starts wearing glasses (Szegedy et al., 2013). The computer only knows what it has seen.

DOI: 10.4324/9781003514237-1

Consider the case involving the way social media entity "X" programmed its automated cropping of images. It was uncovered that X's algorithm had a bias; it disproportionately cropped in favor of individuals with lighter skin tones and women (Zou & Schiebinger, 2018). This issue invoked public criticism and sparked a dialogue on AI fairness and the need for transparent algorithmic practices. It underscored the critical need for precise, impartial data labeling, and the ramifications of deficiencies in these areas. Likely, the computer was trained on images in two forms, photos that were cropped by humans to feature the most interesting part of a photo, and the uncropped version of that photo. By showing the computer how images are typically cropped, the algorithm learned how to crop photos. However, because the labeled data (the cropped photos) likely had a disproportionate number of individuals with lighter skin, the machine-learning algorithm likely inadvertently picked up on skin color as a feature.

As AI evolves, precise, impartial data labeling remains a critical challenge. This is where content analysis comes in. In this book, I will argue that it is particularly helpful in advancing the field of artificial intelligence due to its systematic nature and ability to provide a framework to derive reliable and replicable findings from diverse datasets (Krippendorff, 2013). Content analysis primarily focuses on identifying specific characteristics within content, serving as a powerful means to interpret complex communication patterns, especially within our increasingly digital environment (Neuendorf, 2017).

According to Riffe et al., (2019), content analysis computes and examines the occurrence, connotations, and associations among particular words, themes, or concepts appearing in text or, more broadly, any type of media. This quantitative approach involves sorting information into categories based on systematic rules and then assessing the connections between these categories using statistical measures (Riffe et al., 2019).

Content analysis in the media realm finds its roots in the scrutinization of print materials by the Church in the 17th century. Its development was notably influenced by the 18th-century Swedish debate on the Songs of Zion, a set of hymns that stirred discussions on the interpretation of symbols and textual meanings, a central element of contemporary content analysis (Krippendorff, 2013). Further evolution of content analysis occurred with the explosion of mass-printed newspapers in the early 20th century. Researchers started exploring newspaper coverage on various topics, shedding light on the thematic content and reflecting societal attitudes. This type of analysis was subsequently adapted to additional media formats, including radio broadcasts, television programming, and social media posts (Neuendorf, 2017).

Krippendorff (2022) critiqued early perspectives, such as those advocated by Berelson (1952), by disputing the assumption that content exists as a

manifest element within messages and by questioning the sole reliance on quantitative techniques. Krippendorff (2022) suggests that content emerges through the conceptual engagement of individuals with text, maintaining that textual meanings are not uniform and fixed. He contends that the significance drawn from texts is subjective, engaging with the reader's own context rather than pointing to an inherent meaning within the message itself (Krippendorff, 2022). While there is some scholarly debate over the degree to which humans can classify documents with a high level of objectivity, or whether subjective influences are an inseparable part of certain tasks, this book will adopt Riffe et al. (2019)'s consensus that documents can be objectively labeled, at least in some straightforward cases.

Quantitative content analysis is fundamentally more objective as researchers delineate specific, pre-established categories alongside precise coding protocols, which are then consistently enforced across the content (Krippendorff, 2018). The goal is to diminish the influences of subjectivity and bias, presenting the data in a statistical manner. Conversely, qualitative content analysis leans towards a more interpretive and subjective stance. This method entails a detailed and nuanced scrutiny of the content, with an emphasis on unearthing themes, patterns, and deeper meanings. Here, researchers' perspectives and lived experiences might color their interpretation, thus injecting a measure of subjectivity (Schreier, 2012). Within the confines of this book, we proceed with the presumption that, if content analysis is meticulously defined and the concept is accessible even without an advanced degree, documents can be labeled objectively.

Unitization and Measurement

The concept of unitizing, as defined by Krippendorff (2004), is foundational in content analysis. Imagine dissecting a multifaceted jigsaw puzzle into tinier, more controllable segments. These fractions represent the core and enlightening segments of the content under scrutiny. This is akin to segmenting a film into its constituent scenes or partitioning a novel into chapters. This technique not only highlights the pivotal components but also facilitates a more meticulous and manageable examination process (Krippendorff, 2004).

Selecting appropriate units is essential for effective analysis. This often necessitates the exclusion of extraneous or non-essential information to concentrate on the data that directly pertains to the research question. For example, in the analysis of a political speech, an analyst might prioritize the evaluation of substantive statements and arguments, disregarding peripheral elements such as introductory greetings or informal banter (Krippendorff, 2018). Furthermore, an analyst may sometimes categorize the speech content more broadly rather than analyzing each proposition, categorizing

statements as either "positive" or "negative" to provide a more holistic view of the speech's sentiment (Neuendorf, 2017).

Krippendorff's model for content analysis suggests that a researcher should remain flexible and willing to adapt their methodology to align with the emerging patterns within the data (Krippendorff, 2018). As you delve into the data, your initial hypotheses may evolve. For instance, you may begin by exploring persuasive strategies in speeches, but discover that emotive language is more predominant, prompting a shift in focus to emotional rhetoric instead. Such adaptability is vital; it allows the data to steer your analytical process. The essence of unitizing, and content analysis broadly, is the deciphering of data effectively and dependably.

When undertaking content analysis, researchers are engaging in systematic data measurements. These measurements aim to collect information on variables pertinent to the research question (Neuendorf, 2017). There are two principal concerns when it comes to measurement: reliability and validity. Reliability refers to the consistency with which data can be labeled, and validity addresses the accuracy and truthfulness of the measurement in relation to the intended construct or concept (Riffe et al., 2019).

Reliability is central to the trustworthiness of measurement outcomes in any research. For content analysis, it poses the question of whether repeating a particular method of analysis would yield consistent results. Essentially, the reliability of a measure indicates the extent to which it is without bias and ensures stable measurement across different occasions and observers (Riffe et al., 2019). If you repeated your content analysis, would you get similar results? A reliable content analysis yields similar results, even if different individuals analyze the same content, and even when the same analysis is conducted at separate times.

Your content analysis must also include a sample of documents that is *representative* of what you are claiming to study. As a researcher interested in understanding the public's perspective on climate change, you might consider examining TikTok videos as a barometer of public opinion. This approach is based on the premise that the content of such videos could serve as a proxy for societal sentiments. However, it is essential to recognize that TikTok has a predominantly younger user base that is adept at navigating the online world, potentially biasing the results. This limitation is significant since certain segments of the population, such as the elderly, individuals who lack internet access, or those who simply do not engage with TikTok, may not be sufficiently represented in your analysis, leading to skewed data.

As such, TikTok data can offer a glimpse into the perspectives of certain individuals on the subject of climate change but should not be mistaken for an all-encompassing reflection of the general public's standpoint. A robust assessment would require scrutiny of the extent to which TikTok content—subject to the sway of algorithms, prevailing trends, and singular content

producers—authentically embodies the wider populace's feelings (Highfield & Leaver, 2015). At this juncture, the soundness of your evaluative metric is potentially compromised, for the material under review may not efficiently encapsulate the more extensive subject that piques your interest (Burgess & Green, 2018).

Maintaining an awareness of both methodological soundness and contextual understanding is critical in enhancing the validity of content analysis outcomes (Krippendorff, 2018). It is paramount to acknowledge that, notwithstanding the rigor of the methodology employed, interpretation necessitates a human touch—a deep comprehension of the contextual intricacies, a solid grasp of the respective field, and the application of informed discernment (Maxwell, 2020). These elements elevate the role from being simply an analyst of data to a genuine scholar in the research sphere (Neuendorf, 2016).

Sampling in the Age of AI

The notion of sampling is closely interconnected with the constructs of reliability and validity (Kumar, 2019). Specifically within content analysis, the primary concern lies not only in the nature of the content under investigation but also in the segments of content selected for scrutiny, which is where the role of sampling is emphasized (Bryman, 2016).

To extend the earlier example regarding TikTok, imagine the challenge of meticulously examining every TikTok video that covers climate change. Given the overwhelming number of videos uploaded daily, such a task is practically unfeasible. To render this project feasible, one would need to employ a methodological approach by selecting a sample—a smaller, but representative, collection of TikTok videos—for analysis (Bryman, 2016).

The sampling method deployed influences the validity of research findings. Selecting a sample at random can bolster the likelihood of it being representative of the broader population, thereby mitigating potential biases (Vogt, 2019). By contrast, a non-random sampling strategy, such as selecting TikTok videos based exclusively on their popularity—measured through views or likes—limits the sample to reflecting prevailing opinions and may result in a skewed portrayal of the subject matter (Sue & Ritter, 2012). That said, targeting viral content specifically might align with a researcher's deliberate focus on phenomena with widespread influence. Recognizing and delineating these methodological choices is vital from the outset to ensure precise and consistent terminology throughout the research documentation process. Ultimately, the crux of proficient sampling is ensuring that the sample accurately mirrors the entire population under study (Babbie, 2020).

AI and machine-learning technologies have significantly alleviated concerns related to sampling processes. Ideally, we can employ a computational

system to exhaustively analyze the plethora of TikTok videos relating to climate change, thus obviating the necessity for sampling (Hastie et al., 2009). However, if we rely on AI to label all the data, the most meticulous and painstaking step is verifying that the data is indeed labeled correctly and resembles a human approach to data categorization (Russell & Norvig, 2016). The paradox lies in the inception of this training process, which inherently demands the collection of a representative sample (Bishop, 2006).

"Training data" is pivotal in machine learning, serving as a benchmark for expected outcomes and as such guiding computational models. In computer science vernacular, such datasets may be referred to as "gold standard data," essentially signifying the optimal performance we aim for the computer to emulate (Brookshear & Brylow, 2014). In *supervised* machine-learning tasks human experts are required to manually label and scrutinize a subset of data—this forms what is known as the training set. Subsequently, the algorithm uses this curated dataset as a basis for learning, enabling it to categorize and examine new, unseen data (Alpaydin, 2020). A critical inquiry arises in this methodology: How extensive must the volume of data manually labeled by humans be for the machine to independently continue the task (Halevy et al., 2009)?

The question of when to allow automated systems to take over tasks that require human judgment is intricate and varies widely across different fields and specific activities. As we explore in this book, determining the optimal balance (Brynjolfsson & McAfee, 2014) between human effort and machine automation is the goal. Successfully identifying this equilibrium can significantly enhance the scalability and precision of your content-analysis processes (Kitchin, 2014).

A Content-Analysis Case Study: Contextual Advertising

The sheer amount of content generated by news organizations daily is staggering. Every news piece, broadcast segment, or update on a social platform presents an abundance of data awaiting analytical exploration. Imagine you're an advertiser looking to put your ad alongside some of this content, specifically news about climate change. Content analysis plays a pivotal role in programmatic contextual advertising, which typifies the modern advertising landscape—a behemoth propelled by data and guided by black-box algorithms. At the heart of this sector is the examination of content where countless pieces of news media are systematically evaluated, classified, and matched with relevant advertisements. Such operations represent an ongoing, extensive application of content analysis specific to the assessment of news articles.

For instance, consider how a digital publication detailing contemporary smartphone advancements is evaluated by a sophisticated advertising system. Such a system would dissect the primary subjects within the text—mobile technology, cutting-edge developments—and then strategically match the content with corresponding adverts, like accessories for phones, cellular service offers, or the latest tech devices. Despite the procedure's outward simplicity, it's supported by some type of computational algorithm.

Martin Porter, known for inventing the Porter Stemmer algorithm and the Snowball programming framework, co-cofounded Grapeshot, a company specializing in contextual targeting and content recommendation. He brought his expertise in linguistics and computer science to the development of Grapeshot's technology. The company's approach involves analyzing the text on webpages and identifying key themes and words through a method that resembles the bag-of-words model. This allows for the extraction of keywords without considering the order or contextual usage of words, focusing instead on their frequency and relevance. Grapeshot's success in the ad tech market was significant, and it attracted the attention of Oracle Corporation. The sale of Grapeshot to Oracle was a notable event in the tech industry, with the company selling for $325 million (Pirrone, 2018).

Despite the hefty price tag, most classification systems that work at scale fail to provide conventional benchmarks for external validity or reliability, as we might expect in content analysis. *External validity* is the extent to which data-labeling processes can generalize to different scenarios, environments, or demographic groups, while, as mentioned, reliability relates to consistency (Joppe, 2000; Trochim & Donnelly, 2008). This involves ensuring that each article's classification is accurate, consistent, and representative of reality. Unfortunately, these key measures are often compromised or overlooked in the categorization approach of contextual advertising systems, leading to imprecise or misleading ad placement.

I encourage you to think critically about the limitations of a system that only looks for words and ignores usage and context. For instance, a system that relies solely on keywords or phrases to label posts may fail to take into account the wider semantic context. A discussion critiquing the adverse effects of fast food could be incorrectly tagged as endorsing it if certain buzzwords such as "tasty" and "fast casual" are detected. This might seem like a harmless misclassification but consider the concept of negation. If an article says, "Nothing in this fast casual restaurant could be perceived as tasty," the system would position a fast-food chain's advertisements alongside the critical content (Ferrara et al., 2016). Such inaccuracies, particularly if they occur frequently, can dilute the effectiveness of an advertising initiative and damage the trust in the advertiser's brand (Goldsmith & Lafferty, 2002).

If advertisers better understood content analysis and its connection to AI, they could ask questions and obtain data from contextual advertising vendors that could help them empirically verify the algorithms they use. This sunlight might inspire companies in advertising technology to devise sophisticated, targeted, and genuinely effective classification methodologies. I encourage any advertiser to manually review the classifications made by their vendors for this very reason.

Could advertisers label every news article, social media post, and YouTube video they could appear alongside like a detailed researcher could? Of course not. The world needs computational capabilities to sift through and interpret extensive data sets (Krippendorff, 2013). As such, modern research frequently incorporates computer-facilitated content analysis and social media data, exemplifying the method's versatility and continued evolution (Bryman, 2016).

Take, for example, an inquiry into the portrayal of women in news media spanning the last ten years. A researcher could embark on a manual review, coding a sample of content individually—a task potentially stretching over years. Alternatively, computational tools and machine-learning algorithms could be applied to automate the categorization, leading to a more efficient and scalable content-analysis process. Of course, there are compromises, such as the potential loss of finer qualitative nuances. However, if the researchers apply the core teachings of content analysis, they can still show in their research that they have reached external validity and that they have coded their data accurately and precisely. In the coming chapters, we'll walk through various approaches to save you time when wading through and labeling large amounts of data.

Language as Inferences to Societal Norms

In content analysis, drawing conclusions—known as inferences—from data is paramount. This involves interpreting the outcomes to discern trends or recurring patterns. A researcher might, for instance, conclude that the portrayal of women in passive roles within news content is indicative of broader societal biases (Krippendorff, 2013). Coding or segmenting content into discrete units follows, where news portrayals might be classified based on the depicted roles of women (passive, active, neutral), laying the groundwork for systematic analysis (Neuendorf, 2017). Categorization entails organizing the coded data into relevant groups for analysis, such as sorting content by the dominant depiction of women's roles. These groupings provide a structured framework for data interpretation (Weber, 1990).

The utility of content analysis spans beyond the examination of linguistic patterns—it can probe the framing effects of language and its capacity to construct and reshape realities through various linguistic constructs and

cultural narratives (Altheide, 2013). For example, take the discussion on climate change that we see in the media. Various news outlets may present the issue using different language strategies. Certain reports might opt for language that evokes fear, underlining the disastrous consequences of climate change and the critical need for prompt action (Hansen & Machin, 2013). These editorial choices can shape public perception of climate change, depicting it as an immediate and dire emergency that evokes a deep sense of concern and urgency among the audience (O'Neill & Nicholson-Cole, 2009).

Alternatively, certain websites may present a more detached or even skeptical attitude, minimizing the perceived dangers tied to climatic shifts. Terminology such as "so-called climate change" or "purported global warming" can insidiously inject skepticism concerning the existence and seriousness of climate change. Such wording has the potential to influence individuals' perceptions, causing them to view climate change as a non-critical issue or one open to debate, thereby crafting an altered perception of reality (Corner et al., 2012).

Content analysis within linguistics can be utilized to examine how texts contribute to the construction of social norms. Such literature frequently perpetuates gender stereotypes, albeit in a subtle fashion (Weitzman et al., 1972). For instance, a quintessential fairy tale might portray princesses using attributes such as beauty and fragility, often cast in a passive role awaiting rescue. Conversely, princes are depicted as embodiments of courage, strength, and proactivity (Baker-Sperry & Grauerholz, 2003). These narrative choices have the potential to influence children's understanding of gender roles, implicitly suggesting that femininity is associated with attractiveness and passivity whereas masculinity is linked with heroism and vigor (Hamilton et al., 2006).

In both illustrated cases, the influence of language in framing perception becomes clear. Investigating these linguistic patterns via content analysis enables scholars to unearth and address latent prejudices, thereby promoting a deeper, more enlightened dialogue on articulating intricate subjects such as climate change and gender roles with democratic fairness.

As Krippendorff (2022) underscores, analyzing the conceptual framework that a text constructs for its audience is fundamental. A text does not merely transmit a narrative; it plays an active role in constructing realities for those who engage with it. The significance of scrutinizing linguistic representations carefully and systematically is instrumental in deciphering the influence of language and contributing to a more enlightened society.

When conducting scholarly research, one must precisely delineate the concepts to be examined. Such concepts inevitably shape the research hypotheses and influence the trajectory of the investigation. The selection of these concepts should be reasoned and anchored in robust theoretical

frameworks, existing scholarly work, and the specific research questions being addressed (Denscombe, 2014).

Picking the Right Concepts and Coding Scheme

Underlying concepts operate as a foundational framework for your research, playing a crucial role in encapsulating and quantifying the subtleties of the phenomena or attributes under scrutiny (Miles & Huberman, 1994). Take, for instance, the examination of electronic word-of-mouth (eWOM) advertising. Researchers might be particularly interested in the concept of "sentiment," which encapsulates the positive or negative emotions expressed in customer feedback regarding a product or brand (Liu, 2006).

Sentiment analysis, often utilized in eWOM research, serves as a barometer for assessing public opinion. For example, a predominance of positive sentiment within online reviews often correlates with enhanced public perception, which can subsequently contribute to a boost in both product popularity and sales (Hennig-Thurau et al., 2004). Its applications extend to aiding marketers in the appraisal of their promotional strategies and quantifying the success of such initiatives (Liu, 2006).

To ensure academic integrity and adhere to scholarly standards, we must refrain from over-relying on predetermined constructs such as "sentiment" when they do not precisely align with the research concept at hand. Opting for such constructs out of convenience or prevalence could undermine the specificity and relevance of the investigation. When handling intricate or specialized concepts, it becomes crucial to delineate tailored constructs that are deeply rooted in the related theoretical framework and scholarly discourse (Shields & Rangarjan, 2013).

In exploring the impact of eWOM advertising on consumer purchasing choices, it might be insufficient to focus solely on the "sentiment" of reviews. While positive feedback can suggest a satisfactory product experience, the process behind a customer's decision to purchase can be profoundly multifaceted. This complexity encompasses elements such as the credibility of the reviewer (Metzger et al., 2010), the concept of social proof where potential buyers are influenced by the actions and approvals of others (Cialdini, 2001), the perceived usefulness of the product, and the degree to which an individual is influenced by their peers (Goldsmith & Clark, 2008).

In cases where analysis is centered solely around "sentiment," there is a risk of overly simplifying and misconstruing the data. This simplification may lead to skewed results, directing the research trajectory off course. Sentiment analysis does not capture the complexities of trust or social validation, both of which are fundamental in understanding eWOM (Kang & Johnson, 2013).

Rather than relying on broad and overused constructs such as sentiment, I encourage you to dedicate effort to developing specific and relevant constructs that accurately reflect the phenomena under investigation. Taking the instance of eWOM in advertising studies, constructs that are more targeted—such as reviewer credibility, information quality, and peer influence (e.g., De Vries et al., 2017; Cheung & Thadani, 2012)—might prove to be significantly more insightful and appropriate.

The nuanced and precise delineation and development of analytical frameworks are pivotal for unlocking the depth of your research discoveries. This endeavor necessitates robust theoretical knowledge, careful attention to detail, and profound reverence for the data involved (Babbie, 2016). Indeed, the dimensions of investigation that are quantified become the focal points of your scholarly work. It is, therefore, incumbent upon you, the researcher, to verify that these measurements replicate actuality with maximal fidelity and authenticity (Mertens, 2014).

Imagine online evaluations of a specific smartphone brand via a collection of Amazon reviews. The variability within the research objective may relate to the array of features highlighted by reviewers—such as display clarity, operating system efficiency, battery durability, photographic functions, and so forth. When devising your coding scheme, it must encompass categories that sufficiently represent these distinct features. Should your coding merely register undifferentiated favorable or unfavorable impressions, it is likely that the nuanced distinctions within the research subject are not fully captured, leading to a significant loss of vital details.

A better approach would factor in the potential variations that exist within the different attributes under examination. For example, when evaluating screen quality, one must consider a variety of factors, including resolution, brightness, and color accuracy, to accurately reflect the complexity of these aspects. Effective instructions for human annotators ought to include these subtleties to provide a comprehensive analysis.

Suppose your focus of study is the partisanship reflected in media reporting. The scope of this research is remarkably broad due to the multitude of expressions that partisan perspectives may take. These expressions range from the selection of topics to the descriptive choices framing particular events or figures (Entman, 1993). To precisely assess bias, the research methodology must incorporate coding protocols that capture the subtleties across these facets. For instance, quantifying bias in the selection of subjects could include tracking the relative frequency with which certain topics are reported compared to others (D'Alessio & Allen, 2000). Meanwhile, uncovering linguistic partiality might demand pinpointing the use of certain adjectives that denote a slant when describing individuals affiliated with divergent political ideologies (Somasundaram & Vavreck, 2010).

Hence, it is crucial to adopt a multifaceted metric when analyzing bias, which transcends the simplistic dichotomy of "positive or negative." A nuanced strategy is necessary to uncover and scrutinize the complexities and subtleties inherent in the phenomena being studied.

On to Computational Analysis

The meticulous development of your analytical strategy, specifically the instructions you give to human annotators, is imperative for ensuring the strength and dependability of your content-analysis outcomes (Krippendorff, 2018). In other words, the instrument you employ must be well-defined and generalizable to the entire dataset to be annotated. This is the only way you can aptly uncover the intricate details inherent in your research objective (Neuendorf, 2017). In upcoming chapters, you will build algorithms that classify things based on these concepts and strategies. While doing so, I encourage you to recognize that your "algorithms" are specific to the analytical constructs developed at the onset and may not generalize to other circumstances.

This will prove frustrating. Your TikTok topic classification algorithm may not classify Reddit discussions well. It may not even classify TikTok content well a few months later. This is due to the inherent dynamism of language—its continuous evolution and shift over time invariably degrade the performance of your algorithm (Kearns & Vazirani, 1994). An algorithm's inception marks the peak of its performance, with a gradual decline thereafter (Russell & Norvig, 2016).

In this book, we'll start with the research aims of various computational content-analysis studies that I've published or ones published by colleagues. We embark on constructing algorithms aimed at classification tasks paying close attention to the analytical frameworks that were devised, the codebooks and definitions used, and the algorithms that resulted. Provided you know a bit of Python, or are using a computer-assisted coding tool such as Julius.ai, I am confident you can use content analysis alongside computational tools to systematically analyze communication data and make sense of our increasingly digital world.

References

Alpaydin, E. (2020). *Introduction to machine learning* (4th ed.). MIT Press.

Altheide, D. (2013). Media logic, social control, and fear. *Communication Theory*, *23*(3), 223–228.

Babbie, E. (2016). *The practice of social research* (14th ed.). Cengage Learning.

Babbie, E. (2020). *The basics of social research.* Cengage Learning.

Baker-Sperry, L., & Grauerholz, L. (2003). The pervasiveness and persistence of the feminine beauty ideal in children's fairy tales. *Gender & Society*, *17*(5), 711–726. https://doi.org/10.1177/08912432032556

Berelson, B. (1952). *Content analysis in communication research*. Free Press.
Brookshear, G., & Brylow, D. (2014). *Computer science: An overview* (12th ed.). Pearson
Cialdini, R. B. (2001). *Influence: Science and practice* (4th ed.). Allyn & Bacon.
Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.
Bryman, A. (2016). *Social research methods* (5th ed.). Oxford University Press.
Brynjolfsson, E., & McAfee, A. (2014). *The second machine age: Work, progress, and prosperity in a time of brilliant technologies*. W.W. Norton.
Burgess, J., & Green, J. (2018). *YouTube: Online video and participatory culture*. Polity.
Cheung, C. M. K., & Thadani, D. R. (2012). The impact of electronic word-of-mouth communication: A literature analysis and integrative model. *Decision Support Systems*, *54*(1), 461–470. https://doi.org/10.1016/j.dss.2012.06.008
Corner, A., Whitmarsh, L., & Xenias, D. (2012). Uncertainty, scepticism and attitudes towards climate change: Biased assimilation and attitude polarisation. *Climatic Change*, *114*(3–4), 463–478. https://doi.org/10.1007/s10584-012-0424-6
D'Alessio, D., & Allen, M. (2000). Media bias in presidential elections: A meta-analysis. *Journal of Communication*, *50*(4), 133–156. https://doi.org/10.1111/j.1460-2466.2000.tb02866.x
De Vries, L., Gensler, S., & Leeflang, P. S. H. (2017). Effects of traditional advertising and social messages on brand-building metrics and customer acquisition. *Journal of Marketing*, *81*(5), 1–15. https://doi.org/10.1509/jm.15.0178
Denscombe, M. (2014). *The good research guide: For small-scale social research projects*. Open University Press.
Entman, R. M. (1993). Framing: Toward clarification of a fractured paradigm. *Journal of Communication*, *43*(4), 51–58. https://doi.org/10.1111/j.1460-2466.1993.tb01304.x
Ferrara, E., Varol, O., Davis, C., Menczer, F., & Flammini, A. (2016). The rise of social bots. *Communications of the ACM*, *59*(7), 96–104. https://doi.org/10.1145/2818717
Goldfarb, A., & Tucker, C. E. (2011). Privacy regulation and online advertising. *Management Science*, *57*(1), 57–71. https://doi.org/10.1287/mnsc.1100.1246
Goldsmith, R. E., & Clark, R. A. (2008). An analysis of factors affecting fashion opinion leadership and fashion opinion seeking. *Journal of Fashion Marketing and Management*, *12*(3), 308–322. https://doi.org/10.1108/13612020810889272
Goldsmith, R. E., & Lafferty, B. A. (2002). Consumer response to websites and their influence on advertising effectiveness. *Internet Research*, *12*(4), 318–328.http://doi.org/10.1108/10662240210438407
Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.
Hamilton, M. C., Anderson, D., Broaddus, M., & Young, K. (2006). Gender stereotyping and under-representation of female characters in 200 popular children's picture books: A twenty-first century update. *Sex Roles*, *55*(11–12), 757–765. https://doi.org/10.1007/s11199-006-9128-6
Halevy, A., Norvig, P., & Pereira, F. (2009). The unreasonable effectiveness of data. *IEEE Intelligent Systems*, *24*(2), 8–12. https://doi.org/10.1109/MIS.2009.36
Hansen, A., & Machin, D. (2013). *Media and communication research methods*. Palgrave Macmillan.
Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction*. Springer. https://doi.org/10.1007/978-0-387-84858-7
Hennig-Thurau, T., Gwinner, K. P., Walsh, G., & Gremler, D. D. (2004). Electronic word-of-mouth via consumer-opinion platforms: What motivates consumers to

articulate themselves on the Internet? *Journal of Interactive Marketing*, *18*(1), 38–52. http://doi.org/10.1002/dir.10073
Highfield, T., & Leaver, T. (2015). A methodology for mapping Instagram hashtags. *First Monday*, *20*(1–5). https://doi.org/10.5210/fm.v20i1.5563
Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260.
Joppe, M. (2000). *The research process*. https://www.uoguelph.ca/hftm/research-process
Kang, J.-Y. M., & Johnson, K. K. P. (2013). How does social commerce work for apparel shopping? Apparel social e-shopping with social network storefronts. *Journal of Customer Behaviour*, *12*(1), 83–100. https://doi.org/10.1362/147539213X13645550618524
Kearns, M., & Vazirani, U. (1994). *An introduction to computational learning theory*. MIT Press.
Kitchin, R. (2014). *The data revolution: Big data, open data, data infrastructures and their consequences*. Sage. https://doi.org/10.4135/9781473909472
Krippendorff, K. (2004). *Content analysis: An introduction to its methodology* (2nd ed.). Sage.
Krippendorff, K. (2013, 2018, 2022). *Content analysis: An introduction to its methodology*. Sage.
Kumar, R. (2019). *Research methodology: A step-by-step guide for beginners* (5th ed.). Sage.
LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539
Liu, B. (2006). Sentiment analysis and subjectivity. In N. Indurkhya & F. J. Damerau (Eds.), *Handbook of natural language processing* (pp. 627–666). Chapman & Hall/CRC.
Maxwell, J. A. (2020). *Qualitative research design: An interactive approach*. Sage.
Metzger, M. J., Flanagin, A. J., & Medders, R. B. (2010). Social and heuristic approaches to credibility evaluation online. *Journal of Communication*, *60*(3), 413–439. https://doi.org/10.1111/j.1460-2466.2010.01488.x
Mertens, D. M. (2014). *Research and evaluation in education and psychology: Integrating diversity with quantitative, qualitative, and mixed methods* (4th ed.). Sage.
Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook*. Sage.
Neuendorf, K. A. (2016). *The content analysis guidebook*. Sage.
Neuendorf, K. A. (2017). *The content analysis guidebook* (2nd ed.). Sage.
O'Neill, S., & Nicholson-Cole, S. (2009). "Fear won't do it": Promoting positive engagement with climate change through visual and iconic representations. *Science Communication*, *30*(3), 355–379. https://doi.org/10.1177/1075547008329201
Pirrone, G. (2018, April 27). *Oracle takes a "shot" & acquires brand safety platform, Grapeshot*. CMS-Connected. https://www.cms-connected.com/News-Archive/April-2018/Oracle-Takes-A-Shot"-Acquires-Brand-Safety-Platform,-Grapeshot
Riffe, D., Lacy, S., & Fico, F. (2019). *Analyzing media messages: Using quantitative content analysis in research* (4th ed.). Routledge.
Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: A modern approach* (3rd ed.). Pearson.
Schreier, M. (2012). *Qualitative content analysis in practice*. Sage.
Shields, P. M., & Rangarjan, N. (2013). *A playbook for research methods: Integrating conceptual frameworks and project management*. New Forums Press.
Somasundaram, N., & Vavreck, L. (2010). The power of political adjectives: Measuring the influence of descriptive labels on political perceptions. *Political Communication*, *27*(4), 394–412.

Sue, V. M., & Ritter, L. A. (2012). *Conducting online surveys* (2nd ed.). Sage.
Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). *Intriguing properties of neural networks.* arXiv. https://doi.org/10.48550/arXiv.1312.6199
Taigman, Y., Yang, M., Ranzato, M. A., & Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2014)* (1701–1708).
Trochim, W. M., & Donnelly, J. P. (2008). *The research methods knowledge base* (3rd ed.). Atomic Dog.
Vogt, W. P. (2019). *Dictionary of statistics & methodology: A nontechnical guide for the social sciences* (5th ed.). Sage.
Weber, R. P. (1990). *Basic content analysis* (2nd ed.). Sage.
Weitzman, L. J., Eifler, D., Hokada, E., & Ross, C. (1972). Sex-role socialization in picture books for preschool children. *American Journal of Sociology*, *77*(6), 1125–1150.
Zou, J., & Schiebinger, L. (2018). AI can be sexist and racist—it's time to make it fair. *Nature News, 559*(7714), 324–326. https://doi.org/10.1038/d41586-018-05707-8

Chapter 2

Designing a Computational Content Analysis

An Illustration from "Civic Engagement, Social Capital, and Ideological Extremity"

As we've covered thus far, content analysis offers the researcher a structured method to interpret and label data. It presents a systematic approach to decoding textual data, which is essential in maintaining research rigor. Effective content analysis is grounded in two fundamental principles: systematic procedures and validity (Krippendorff, 2018). Before we set off to do innovative computational analysis, let's first take time to be intentional and premeditated in what we're going to measure and operationalize it in a way that will pass muster for academic research. This chapter illustrates best practices to accomplish these tasks, through the lens of one of my favorite content analyses that I conducted a few years ago, "Civic Engagement, Social Capital, and Ideological Extremity: Exploring Online Political Engagement and Political Expression on Facebook" (Ferrucci et al., 2020). I close the chapter with some thoughts on diversity in content analysis, and provide you with a few tools to help with the data-labeling process.

Thinking Intentionally about Operationalization

Operationalization is a frequently neglected step in the research methodology that entails the clear definition of variables intended for measurement. By operationalizing variables, researchers ensure that they move from theoretical constructs to quantifiable entities grounded within empirical observation. The rigor of this process is often a determinant of a study's success, as it is instrumental in aligning collected data with the targeted conceptual framework. Without meticulous attention to operationalization, the risk of gathering data that inadequately represents the investigated concepts increases significantly. Put another way, we need to meditate on precisely what it is we're going to measure before we actually do so, or else we run the risk of what I call "concept drift," which is a study that doesn't exactly measure what it ought to. Reviewers are good at spotting this oversight, and yes, reviewer #2 will hate it.

DOI: 10.4324/9781003514237-2

In the aforementioned paper, we explored Facebook as a venue for political expression. As someone with a passion for methodology, I have often found conventional survey methods asking about low-frequency and unmemorable activities to be inadequate. It's a fact that most people don't frequently make political posts on social media, and alongside all of the content an individual generates on Facebook, it's likely beyond their immediate ability to recall how much political content they generate (Tucker et al., 2018). Given this, inaccuracies likely arise in our understanding of online content sharing due to both numeracy and social desirability biases (Larson, 2019). To overcome these challenges, we crafted a methodological strategy that integrated participants' self-reported survey data with their behavioral data on Facebook. This allowed for a direct comparison of self-reported political engagement against actual political postings by that user on Facebook (Macke et al., 2022). After obtaining consent from our study participants, they permitted a specialized application to collect their Facebook activity. We employed the Facebook Graph API to gather a range of content from each participant's profile (Ferrucci et al., 2020). Our approach was indeed reminiscent of the technique used by the infamous Cambridge Analytica. However, our research adhered to ethical standards: securing IRB approval, anonymizing the data, and ensuring its protection. Furthermore, our intention was not to manipulate elections but to advance academic understanding of political engagement.

Unitizing: The Essential Methodological Approach

Before we can measure "how political someone is on Facebook," we need to think specifically about how that behavior manifests and focus first on exactly what the unit of analysis is important. In content analysis, the process of unitizing, which refers to identifying the basic elements for analysis, is the start (Krippendorff, 2018). It can involve texts, words, phrases, sentences, themes, or, in the case of our study, individual Facebook posts. We decided on Facebook posts as our unit of analysis because they could be efficiently analyzed based on their content and context. This example underscores the importance of ensuring that the units of analysis are both relatable to the research question and commensurate with the characteristics of the data. What else might we have done? We could have looked at what pages they liked, but that would have been more of a passive measure of political engagement. Just because someone likes a Facebook page doesn't mean they are actively participating in politics. Other sources, such as direct messages, might have been even better but we deemed that too personally invasive.

The use of social media platforms, such as Facebook, can serve as a window into user behavior and preferences. The regularity with which

individuals update their statuses or posts could be indicative of their engagement level with the platform. The variety of shared content, including articles, multimedia, and personal photographs, may reveal insights into a user's hobbies, interests, and even political orientations (Vargo et al., 2014). Moreover, the interactive elements of these platforms such as the number of likes, comments, and shares one accumulates can be reflective of a user's tendency to engage with content and potentially their propensity for involvement in digital discourse. The specific timing of social media activity can reveal intricate details about user engagement. Posting during significant political happenings or amid contentious debates may indicate a higher level of political involvement compared with those who remain silent during these times. An analysis that identifies the use of politically charged language in a person's postings can be an indicator of their political interests.

While an assortment of digital footprint information can be gathered from a user's visible activity on social media platforms like Facebook, let's acknowledge the inherent limitations of this data. A user's comprehensive interaction with political content on Facebook cannot be accurately surmised from their public postings alone (Crawford & Finn, 2015). Activities such as silently browsing or privately responding to posts, the nature of the content they elect to read without any direct engagement, or the duration spent perusing specific political material comprise significant elements of an individual's political involvement on the platform. These facets, regrettably, remain beyond the reach of researchers who only analyze publicly shared posts (Tufekci, 2014). As such, it's a major limitation that we had to accept and work around. We were not measuring political engagement or interest but instead how vocal someone was about politics.

The crux of the matter is that existing methodological approaches primarily focus on "active" behaviors or overt interactions, such as those that create a digital footprint like click-throughs, responses, or content creation. Contrastingly, "passive" activities—a term for when users read, scroll, or simply watch content without direct engagement—may also significantly shape political perspectives and actions, yet these do not produce observable traces for analysis (Thorson et al., 2020; Zhang & Pentina, 2012). This dichotomy spotlights the intricacies of examining political participation on social media and highlights the pressing need to refine our research methods. When considering social media trace data, here are a few different units of analysis we could decide to measure:

1. **Posts**: Examining the hashtags, keywords, themes, and/or issues users employ in the content they generate or share on their profile, as a part of their "timeline" or "news feed" is the common choice for studying social media behaviors. This data is often what is most readily available to researchers, and it can uncover trends in political discourse (Murphy et al., 2014).

2. **Comments**: Instead of concentrating solely on the original posts made by users, comments provide a deeper understanding of user engagement with political content; comments are fertile grounds for rich discussions and debates that may not be as apparent in the initial post. Yet, this approach presents practical difficulties—the overwhelming number of comments presents a significant hurdle, introducing a high level of labor intensity and a time-consuming analysis process (Diakopoulos & Naaman, 2011). Furthermore, there is the risk of detachments from the original context when comments are analyzed as standalone fragments.
3. **Likes, follows, and other platform engagement**: Assessing the volume of likes and shares a given user dedicates to specific content or pages offers an alternative approach to quantifying an individual's involvement with politics. If an individual engages regularly with political content by "liking," or "upvoting" the content, it is a signal of their support. Additionally, if we analyze which pages or groups that individuals belong to or "like," we can understand at a higher level what topics they want to see content from. In either case, the algorithms running behind the scenes at social media platforms will likely suggest content they engage with, and in turn this likely means they see political content.
4. **Exposure time**: Analyzing the amount of time users spend viewing certain posts or the frequency at which they are exposed to certain types of content can be insightful. Though it might be hard to gain access to such data outside of conducting a lab-based experiment, this kind of data would provide valuable information about passive engagement with political content, thus offering a more comprehensive analysis.

While alternative unitizing methods may offer more depth or breadth to the analysis, each comes with its own set of challenges and limitations, and the most suitable method will depend on the specific research questions being asked and, perhaps most likely, the data that you have available to you. In our study, we collected the data from both posts and comments. We did not save the likes of individuals, due to a limitation with Facebook's API. Since we weren't running a laboratory experiment, we were left with looking at what users said, not what users consumed or viewed. This fundamental understanding of what we are measuring and what we are not is very important. This limits what we can study, and should directly shape the analysis and literature therein.

Operational Definitions: The Foundation of Content Analysis

When planning the scope of texts or specific units of trace data to be coded, focus your energy on constructing precise operational definitions for the content in question. Allocate the bulk of your effort to developing the

coding scheme, rather than the mechanical aspects of the coding process. From my own scholarly practice, I have observed that failures in content analysis often stem from a lack of rigorously defined and comprehensive guidelines for categorization. For instance, here are a couple of codebooks that inspired us when we were drafting my definition for political talk.

Codebook for Social Media Analysis from "Measuring Networked Social Capital" Study

In their research on social capital in online networks, Ellison et al. (2007) present a robust codebook that outlines distinct categories for user interactions on Facebook. The codebook covers a range of behaviors, from initiating contact and sending messages to the frequency of profile updates and the number of Facebook friends. This instrument shows clear relationships between each category and the overarching research inquiry, aiding in a comprehensive and nuanced analysis of online social capital.

Codebook for Political Engagement on Twitter from "More than a Microblog" Study

Conover et al. (2011) employed a comprehensive codebook to classify different forms of political engagement in their study that investigated patterns of Twitter use in political contexts. The codebook effectively categorized political information sharing, original political content creation, and politically themed retweets. Notably, the codebook also addressed the sentiment of tweets (positive, negative, neutral), lending nuance to their analysis of online political engagement.

Each of these codebooks is an informative example of comprehensive, specific, and reliable measurement tools for content analysis in the research of online behaviors. It's important to remember that unless the concept you're trying to measure is new, there is probably a definition or a combination of definitions available in the extant literature that you can leverage and incorporate into your study. We surveyed the literature and found different types of political talk that commonly occur and created an exhaustive list. While our final label was a binary decision on whether the posts mentioned political talk (0 = no, 1 = yes), by identifying common types and examples of political talk, we were able to ensure that we labeled documents consistently and exhaustively. Political talk was coded as being present if a post had: a reference to a political figure; an exploration or connection to topics of governance and public administration such as taxation, law enforcement, defense, or healthcare; discourse on legislative agendas or actions; commentary on issues of local governance; engagement with prominent societal concerns; allusions to electoral processes or the act of voting; or relevance to judicial

deliberations at a high level, illustrated, for example, by the scrutiny surrounding the Trump administration's travel restrictions often referred to as the "Muslim ban." Even after the literature review, we only felt confident our definition was good after we read several posts from our collected dataset. While an initial literature review uncovered some topics, others were missing, or needed to be adjusted to better describe what we saw in our data.

The process of developing a codebook for content analysis is iterative, necessitating continual refinement and enhancement. It involves crafting code definitions that draw from theoretical frameworks, existing literature, and the researcher's anticipations (Neuendorf, 2016). Still, when coders start working with the actual data, they might stumble upon content elements that resist fitting snugly within these pre-established categories. Referred to as edge cases, these anomalies can significantly challenge the reliability of the coding framework (Riffe et al., 2014). Adaptability becomes crucial when addressing edge cases (Krippendorff, 2013).

After the initial codebook is developed, a preliminary round of coding should be conducted. This first round will highlight any issues, ambiguities, or edge cases not covered by the existing codebook. For instance, there could be forms of expression, nuances in language, or specific contexts not anticipated during the codebook construction (Bazeley, 2013; Saldaña, 2021). These instances serve to not only test the robustness and adequacy of the initial codebook but also provide opportunities for refining it. Following this initial round, researchers should review the cases where coders struggled or disagreed. Regular coder discussions play an essential role in this process. They provide a platform for coders to discuss their challenges, propose possible solutions, and share insights to inform codebook revision. These discussions are instrumental for identifying edge cases and uncovering subtleties or complexities not initially considered (Miles et al., 2019).

Based on these insights, the codebook should be revised to address these issues, either by refining existing codes or adding new ones. Once revisions are made, another round of coding should be conducted, and the codebook's effectiveness reevaluated. This iterative process continues until the codebook reliably captures all nuances of the content and handles edge cases effectively. The coding process is not static but evolves as deeper understandings of the content are gained. It embraces the complexity and diversity of human communication, acknowledging that edge cases are not mere anomalies but valuable data that can offer unique insights. Iterative codebook creation enhances the depth, accuracy, and reliability of content analysis, leading to more robust and valid research findings (Krippendorff, 2018; Schreier, 2012).

Intercoder Reliability

The reality is, no matter how good a codebook or set of rules seems to you, until others can reliably produce similar results given those instructions,

we have no way of empirically knowing if the instructions generalize to the concept we are measuring (Krippendorff, 2018). For these reasons alone it makes sense to take a small sample of documents and have at least two researchers independently label that data and compare their agreement.

Turning back to our study of political talk, after we felt we had a good codebook that handled all the various ways politics could be discussed, my trusted colleague, Toby Hopp, and I independently reviewed a random collection of 100 Facebook posts. We disagreed once ($\kappa = .80$). We talked it out, and it was obvious that one of us missed labeling a political post. That aside, it was clear that the operational definition was working. Now it was time to independently review two random samples. Together we labeled 5,006 (3,937 unique) posts for if they contained political talk. As we'll talk about later in Chapter 4, these annotations were used to build a Keras neural network (Gulli & Pal, 2017).[1]

How to Write Good Content-Analysis Codebooks

Zooming into specific codes, let's review another more myopic example. Imagine a hypothetical codebook using a concept slightly different from political talk—political engagement. We could start with something as simple as:

Code 1: Engaged—User engages with political content.
Code 2: Not Engaged—User doesn't engage with political content.

This version of the codebook is too vague. It does not provide enough instruction on what constitutes "engagement" or differentiate among engagement levels. This can lead to inconsistency in data coding because different coders may have their own interpretations of what "engagement" entails, and what that cutoff is for making someone political or not. Let's try again:

Code 1: Engaged—User actively participates in political discourse by posting original content, sharing political news, commenting on political discussions, or responding to political polls or surveys. Evidence of engagement includes the use of political hashtags, direct mentions of political figures or parties, discussion of policy issues, or advocacy for political action.
Code 2: Not Engaged—User does not exhibit any of the behaviors listed under "Engaged." Posts are devoid of political content, hashtags, mentions of political figures or parties, policy discussions, or advocacy. The user's activity is limited to non-political personal updates, entertainment-related content, or other apolitical interactions.

This revised codebook provides clear, specific guidelines for coders to follow, reducing ambiguity and increasing the likelihood of consistent coding across individuals. It delineates what behaviors and content qualify as

political engagement and what do not, making it easier for coders to make informed decisions.

To further refine the codebook, we could also consider abandoning a binary classification ("political" or "not") and adding more nuanced categories of engagement, such as passive engagement (e.g., liking or viewing political content without commenting or sharing) or differentiating between levels of active engagement (e.g., creating original political content versus simply sharing existing content). Additionally, including examples of each type of engagement and non-engagement can help coders recognize borderline cases.

Let's consider one more example, this time political extremity. Often in political science research, there is a desire to know how extreme one's ideologies are. To that end, we may want to observe social media content from users and adopt the following codebook:

Code 1: Mild Liberal Extremist—User consistently posts political content promoting liberal viewpoints, advocates for liberal policies and politicians, and criticizes conservative ones. However, these actions do not involve personal attacks, misinformation, or the promotion of radical changes in the political system.

Code 2: Extreme Liberal Extremist—User consistently posts political content that aggressively promotes liberal viewpoints, advocates for radical policies and politicians, and strongly criticizes conservative ones. They often use personal attacks, misinformation, and advocate for radical changes in the political system.

Code 3: Mild Conservative Extremist—User consistently posts political content promoting conservative viewpoints, supports conservative policies and politicians, and criticizes liberal ones. However, these actions do not involve personal attacks, misinformation, or the promotion of radical changes in the political system.

Code 4: Extreme Conservative Extremist—User consistently posts political content that aggressively promotes conservative viewpoints, supports radical policies and politicians, and strongly criticizes liberal ones. They often use personal attacks, misinformation, and advocate for radical changes in the political system.

Code 5: Moderate—User posts political content that promotes both liberal and conservative viewpoints, supports both liberal and conservative policies and politicians, and criticizes both. They do not use personal attacks, misinformation, or advocate for radical changes in the political system.

This codebook reads like a first draft. While it encapsulates a lot of interesting ideas, such as mis/disinformation sharing, it is overly complicated and creates a confusing overlap between political ideologies (liberal, conservative) and extremity levels (mild, extreme). It assumes that all political

posts can be categorized as either liberal or conservative, overlooking the likelihood of nuanced posts or those centering on nonpartisan issues. Additionally, using both "mildly extreme" and "extreme" ideological labels can create confusion—they are oxymoronic and may be hard to distinguish in practice. Finally, the subjectivity involved in categorizing "misinformation" and "radical changes" in the political system could lead to inconsistency and bias in coding. This illustrates the importance of keeping codebooks clear, concise, and objective to maximize their practical utility in content analysis. My advice here is to simplify and detach each code. If you want to measure misinformation, do so in a separate code, where a proper definition is provided. Avoid conflating concepts during the initial phases of a content analysis, as it can easily be done later in the data analysis phase.

Case Study: What Is Toxic, Anyway?

A well-constructed codebook is like a meticulously well-organized dictionary, comprehensive in its coverage, clear in its definitions, and concise in its explication. In academia, we have the time, patience, and incentive to get these things right, because it's how we publish science (Krippendorff, 2018). However, in industry, it's not always possible nor desired. Let's go through a concept that I've struggled with in online content moderation—"toxicity"—and talk about how it is defined. Alphabet, the parent company of Google, owns the corporation Jigsaw. Their Perspective tool comes from their broader initiative to spark innovation and help detect/remove awful content like bullying and harassment in various contexts, like YouTube comments (Perspective API, 2021). I applaud them for helping researchers study hate speech and threats online, "at scale" across big datasets (Wulczyn et al., 2017).

Perspective has, however, no formal codebooks that define toxicity. This comes from their inability to enforce one labeling process from their training data. Some data was annotated by comment moderators at the *New York Times*, other data came from crowdsourcing platforms like Amazon Mechanical Turk and Figure Eight (now Appen; Wulczyn et al., 2017). Broadly, this data has been assembled into one broad definition, "a rude, disrespectful, or unreasonable comment that is likely to make you leave a discussion" (Perspective API, 2021). Beyond this, we get a little more detail—the Perspective API can retrieve specific attributes, including:

1. TOXICITY: A rude, disrespectful, or unreasonable comment that is likely to make one leave a discussion.
2. SEVERE_TOXICITY: A very hateful, aggressive, or disrespectful comment that is likely to contribute to a hostile environment.
3. INSULT: Insulting, disrespectful, or bad-natured comments.
4. PROFANITY: The use of intense offensive language.

5. IDENTITY_ATTACK: Negative or hateful comments targeting someone because of their identity.
6. THREAT: Comments that express an intent to hurt someone physically.
7. SEXUALLY_EXPLICIT: Sexually suggestive, detailed, or graphic sexual language or imagery.
8. FLIRTATION: Pickup lines, complimenting appearance, subtle sexual innuendos, etc.

These attributes attempt to parse out the different flavors of toxicity that can exist, but at their core, there are considerable ambiguities. What qualifies as rude or disrespectful? If a post is rude but doesn't make an individual want to stop reading a conversation, does it count? What constitutes a hostile environment? What language is deemed profane and to what extent does a comment need to be negative or hateful to be considered an identity attack? Such questions are inherent in the task of defining and identifying toxicity.

Addressing them, while remarkably challenging, is needed to develop effective measures against online harm and foster healthier digital spaces. Therefore, defining toxicity with clarity and rigor, even in fast-paced industry settings, is not just a desirable goal but a necessary step in the seemingly endless battle against toxic online behavior.

Moderators at Wikipedia likely differ from those at the *New York Times* in what they consider problematic and how they perceive toxicity (Dixon et al., 2018). This means that the underlying definitions that powered this training data are byproducts of the specific content-moderation policies that governed Wikipedia and the *Times*. Let's consider some different ways we could define toxicity:

Code 1: Toxic—The news article uses negative language, contains harmful stereotypes, or promotes division among groups.
Code 2: Non-Toxic—The news article uses neutral language, does not contain harmful stereotypes, and does not promote division among groups.

Terms like "negative language," "harmful stereotypes," and "promotes division" are extremely subjective and can be interpreted differently depending on the coder's perspectives and biases (Sapir, 1929; Whorf, 1956). What one coder might perceive as negative language could be seen as neutral or even positive by another. It depends highly on context, tone, and nuanced understanding of language, none of which this code provides guidelines or definitions for.

Similarly, "harmful stereotypes" and "promoting division" are ambiguous terms that can change depending on cultural, social, and individual perspectives. The lack of precise definitions or examples leaves coders to rely on their own judgment, increasing the risk of inconsistent and biased coding.

Binary labels like "toxic" and "non-toxic" can oversimplify complex issues, leading to difficulties in accurately gauging the nuances of certain texts. Consider the instance where an article might convey harmful stereotypes in an underhanded way, despite using language that appears neutral or even positive on the surface. Alternatively, a report may deliver an impartial account of events, and yet the very incidents it chronicles may unintentionally perpetuate social divisions. In these subtle scenarios, an overly simplistic framework for interpretation falls short of capturing the intricacies involved.

This codebook also lacks concrete operational guidelines or procedures. For example, it doesn't elucidate what specific media should be considered in coding—should the comments that precede a comment in a discussion or comment thread be considered, or must the toxicity be inferred from the context? Let's take all of these into consideration and try again:

INSTRUCTIONS: Code the following comment, and review its context in the greater context of the entire discussion thread.

Code 1: Derogatory Language—The comment uses language that demeans, belittles, or insults a certain group or individual. Examples include aggressive swear words, derogatory slurs, or pejorative nicknames targeting a person or group's race, gender, religion, etc.

Code 2: Stereotyping—The comment presents groups or individuals in a generalized, simplified manner based on their race, nationality, religion, gender, etc. This may include assuming certain characteristics, behaviors, or attitudes as inherent to the group or individual without providing nuanced or balanced perspectives.

Code 3: Unsubstantiated Claims—The comment presents claims that are not backed by reliable evidence or factual information. This may include spreading misinformation, promoting conspiracy theories, or presenting personal opinions as factual statements.

Code 4: Inciting Division—The comment promotes hostility or division among different groups. This may involve framing issues in a polarizing way, inciting hate or fear towards specific groups, or promoting an us-versus-them narrative.

Code 5: Neutral/Non-Uncivil—The comment uses neutral language, presents balanced viewpoints, does not contain harmful stereotypes, does not promote division among groups, and backs claims with reliable evidence or factual information.

The updated codes delineate explicit parameters for identifying behaviors classified as "toxic" within communication, creating a more meticulous instrument for content examination. It manages situations through the subdivision of the generic "toxic" content classification into distinct forms of incivility (Roberts, 2018; Sheth et al., 2022). Consequently,

if a comment demonstrates a single type of incivility without evidencing others, it can be accurately categorized. The granularity and specific detail provided by this enhanced coding manual reduce discrepancies stemming from subjective interpretation among researchers. This then bolsters the consistency of the coding process and fortifies the credibility of the study's findings.

The revision of the broader and somewhat abstract concept of "toxicity" into a more specific notion of "incivility" results in a more targeted and empirical research focus. Concrete parameters for investigation link to specific, existing bodies of literature in the field. Incivility is particularly relevant to several research fields, including political communication, online behavior, cyberbullying, and even interpersonal communication, and now we have directly measured these things. Together, these codes do still represent the umbrella of toxicity. But by focusing on the "incivility" in the comments, researchers can explore and reveal the links between the nature of the articles' content and its potential effect on readers' political attitudes and behaviors, drawing from previous research findings in political communication (e.g., Vargo & Hopp, 2020).

Providing Examples of Codes

To ensure a comprehensive understanding of concepts, providing illustrative examples can be beneficial for annotators as they navigate complex subjects. For instance, consider the concept of uncivil discourse, such as the use of derogatory language. Suppose we encounter a sentence within a comment that reads, "Liberals are totally-divorced-from-reality, and fail to grasp economic truths." This type of language exemplifies the uncivil character of the discourse as it is dismissive and belittling, thereby serving as a vivid example.

In this case, the article uses the phrase "totally-divorced-from-reality" to demean a certain group (liberals). This aligns with Code 1: Derogatory Language, as the language demeans and insults a certain group. Though the edge case here lies in the use of stereotypical language rather than explicit slurs or aggressive insults, the detailed codebook ensures such subtler forms of derogatory language are also captured. Still, it's good to show examples like this so our annotators understand that an insult is an insult.

Next, let's consider stereotyping and the phrase, "Teenagers these days are more interested in TikTok than in meaningful conversation." In this instance, the comment makes a sweeping generalization about all teenagers. Even though this statement might not seem extremely uncivil, it still falls into Code 2: Stereotyping, as it presents an oversimplified view of a group (teenagers). The comprehensive definitions of the codebook make sure such milder, yet still harmful forms of stereotyping are not overlooked.

These edge examples reaffirm the strength and effectiveness of the revised codebook in capturing various forms and levels of incivility. It ensures that even the subtler, milder, or more nuanced instances of incivility do not slip through the cracks, thereby providing a more comprehensive and accurate analysis. When in doubt, include a few examples of each concept you define for your annotators. Do this while keeping in mind that *too many examples* will likely lead to your annotators not reading your instructions closely, and could even generate confusion.

Leveraging Computational Tools for Data Management and Analysis

Once you're ready to start labeling lots of documents, keep in mind that several tools can make your life easier. Here are three top options for labeling media data:

Labelbox (https://www.labelbox.com/) is a collaborative annotation tool that allows teams to label their data efficiently. It supports various types of data, including text, images, and videos. Labelbox offers a user-friendly interface and a range of features, such as bounding boxes, polygons, and semantic segmentation for images and videos. It also supports text classification and named entity recognition for text data. While Labelbox is easy to use, some technical skills may be required to set up and manage the tool effectively. Coding is not required for basic usage, but advanced features may require some programming knowledge.

Doccano (https://doccano.github.io/doccano/) is an open-source text annotation tool that is particularly useful for natural language processing tasks. It supports three types of annotation: text classification, sequence labeling, and sequence-to-sequence annotation. Doccano offers a simple and intuitive interface, making it easy for non-technical users to label their data. However, setting up Doccano requires some technical skills, as it needs to be installed and run on a server. Coding is not required for using Doccano, but some familiarity with command-line interfaces can be helpful in troubleshooting and configuring it.

Label Studio (https://labelstud.io/) is a multi-type data labeling tool that supports text, audio, image, and video data. It offers a wide range of annotation options, including text classification, named entity recognition, object detection, and audio transcription. Label Studio is highly customizable, allowing users to create their own annotation interfaces to suit their specific needs. While using Label Studio does not require coding, setting up and customizing the tool may require some technical skills, including familiarity with HTML and CSS. Label Studio is what we use in several of our studies to label data, and I highly recommend it.

Closing Remarks on Defining Your Content Analysis

The sophisticated algorithms we build in the subsequent chapters all depend on data that has been labeled in a reliable and externally valid way. Don't fall into the toxicity trap I discussed earlier. Strive for clarity, exhaustiveness, and inclusivity in your codes. Operationalize your codes in ways that directly match the definitions of concepts from existing literature whenever possible.

Here is the rubric I use to evaluate my codebooks:

- **Comprehensive**: Ensuring that the codebook covers all relevant aspects of the data (Neuendorf, 2016).
- **Clear**: Providing unambiguous definitions that can be easily understood and applied by coders (Krippendorff, 2018).
- **Consistent**: Applying the same standards across all content to ensure that coding is reliable (Riffe et al., 2014).
- **Concise**: Keeping the codebook brief and to the point to avoid overwhelming coders with unnecessary information (Saldaña, 2021).
- **Contextual**: Considering the context in which content appears, as it can greatly affect the interpretation of the data (Miles et al., 2019).

With these things in mind, start with labeling your data for your content analysis. The following chapters will focus on how we can take labeled data and use computers to label documents for us, under our supervision. From here we will assume that you have a fundamental working knowledge of manipulating data in Python. If you are not a coder, we'll discuss emerging generative AI tools that can help you code. Either way, join me as I unpack the various types of classification tools that are at our disposal today. They will guide you in better understanding and working with computational folks who can help you "scale" your content analysis across large datasets.

Note

1. Data was tokenized and transformed using fasttext.cc (Mikolov et al., 2017). 80% (n = 3,729) of the data was randomly selected as training data, 20% was allotted to test. Training, tuning, and evaluation were performed via Python's ktrain package (Maiya, 2022). The F1 score was .96 for non-political talk and .89 for political talk, suggesting the model was accurate despite biased classes. The log loss was .10 and the validated loss was .20, suggesting the model performed acceptably on unseen data.

References

Bazeley, P. (2013). *Qualitative data analysis: Practical strategies.* Sage.

Conover, M. D., Goncalves, B., Ratkiewicz, J., Flammini, A., & Menczer, F. (2011). Political polarization on Twitter. *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, *5*(1), 89–96. https://doi.org/10.1609/icwsm.v5i1.14126

Crawford, K., & Finn, M. (2015). The limits of crisis data: Analytical and ethical challenges of using social and mobile data to understand disasters. *GeoJournal*, *80*(4), 491–502. https://doi.org/10.1007/s10708-014-9597-z

Diakopoulos, N., & Naaman, M. (2011). Towards quality discourse in online news comments. *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*, *2011*, 133–142. https://doi.org/10.1145/1958824.1958844

Dixon, L., Li, J., Sorensen, J., Thain, N., & Vasserman, L. (2018). Measuring and mitigating unintended bias in text classification. *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, *2018*, 67–73. https://doi.org/10.1145/3278721.3278729

Ellison, N. B., Steinfield, C., & Lampe, C. (2007). The benefits of Facebook "friends": Social capital and college students' use of online social network sites. *Journal of Computer-Mediated Communication*, *12*(4), 1143–1168.

Ferrucci, P., Hopp, T., & Vargo, C. J. (2020). Civic engagement, social capital, and ideological extremity: Exploring online political engagement and political expression on Facebook. *New Media & Society*, *22*(6), 1095–1115.

Gulli, A., & Pal, S. (2017). *Deep learning with Keras: Implementing deep learning models and neural networks with the power of Python*. Packt Publishing.

Krippendorff, K. (2013). *Content analysis: An introduction to its methodology*. Sage.

Krippendorff, K. (2018). *Content analysis: An introduction to its methodology* (4th ed.). Sage.

Larson, R. B. (2019). Controlling social desirability bias. *International Journal of Market Research*, *61*(5), 534–547. https://doi.org/10.1177/1470785318805305

Macke, E., Daviss, C., & Williams-Baron, E. (2022). *Untapped potential: Collecting and analyzing digital trace data within surveys*. SocArXiv. https://doi.org/10.31235/osf.io/frhj6

Maiya, A. S. (2022). ktrain: A low-code library for augmented machine learning. *Journal of Machine Learning Research*, *23*(1), 1–6.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., & Joulin, A. (2017). *Advances in pre-training distributed word representations*. ArXiv. https://doi.org/10.48550/arXiv.1712.09405

Miles, M. B., Huberman, A. M., & Saldaña, J. (2019). *Qualitative data analysis: A methods sourcebook* (4th ed.). Sage.

Murphy, J., Link, M. W., Childs, J. H., Tesfaye, C. L., Dean, E., Stern, M., Pasek, J., Cohen, J., Callegaro, M., & Harwood, P. (2014). Social media in public opinion research: Executive summary of the AAPOR task force on emerging technologies in public opinion research. *Public Opinion Quarterly*, *78*(4), 788–794. http://doi.org/10.1093/poq/nfu053

Neuendorf, K. A. (2016). *The content analysis guidebook*. Sage.

Perspective API. (2021). *Attributes and languages*. https://developers.perspectiveapi.com/s/about-the-api-attributes-and-languages?language=en_US

Riffe, D., Lacy, S., & Fico, F. (2014). *Analyzing media messages: Using quantitative content analysis in research*. Routledge.

Roberts, S. T. (2018). *Behind the screen: Content moderation in the shadows of social media*. Yale University Press.

Saldaña, J. (2021). *The coding manual for qualitative researchers* (4th ed.). Sage.

Sapir, E. (1929). The status of linguistics as a science. *Language*, *5*(4), 207–214.

Schreier, M. (2012). *Qualitative content analysis in practice*. Sage.

Sheth, A., Shalin, V. L., & Kursuncu, U. (2022). Defining and detecting toxicity on social media: Context and knowledge are key. *Neurocomputing*, *490*, 312–318. https://doi.org/10.1016/j.neucom.2021.11.095

Thorson, K., Cotter, K., Medeiros, M., & Pak, C. (2020). Algorithmic inference, political interest, and exposure to news and politics on social media. *Information, Communication & Society*, *23*(2), 223–239. https://doi.org/10.1080/1369118X.2019.1642934

Tucker, J. A., Guess, A., Barberá, P., Vaccari, C., Siegel, A., Sanovich, S., Stukal, D., & Nyhan, B. (2018). *Social media, political polarization, and political disinformation: A review of the scientific literature*. SSRN. https://dx.doi.org/10.2139/ssrn.3144139

Tufekci, Z. (2014). Big questions for social media big data: Representativeness, validity and other methodological pitfalls. *Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media*, *8*(1), 506–514. https://doi.org/10.1609/icwsm.v8i1.14517

Vargo, C. J., Guo, L., McCombs, M., & Shaw, D. L. (2014). Network issue agendas on Twitter during the 2012 U.S. presidential election. *Journal of Communication*, *64*(2), 296–316. https://doi.org/10.1111/jcom.12089

Vargo, C. J., & Hopp, T. (2020). Fear, anger, and political advertisement engagement: A computational case study of Russian-linked Facebook and Instagram content. *Journalism & Mass Communication Quarterly*, *97*(3), 743–761. https://doi.org/10.1177/1077699020911884

Whorf, B.L. (1956). *Language, thought, and reality: Selected writings*. Technology Press of Massachusetts Institute of Technology.

Wulczyn, E., Thain, N., & Dixon, L. (2017). Ex machina: Personal attacks seen at scale. *Proceedings of the 26th International Conference on World Wide Web*, *2017*, 1391–1399. https://doi.org/10.1145/3038912.3052591

Zhang, L., & Pentina, I. (2012). Motivations and usage patterns of Weibo. *Cyberpsychology, Behavior, and Social Networking*, *15*(6), 312–317. https://doi.org/10.1089/cyber.2011.0615

Chapter 3

Basic Information Retrieval for Content Analysis

In a world full of algorithms and constant innovation in AI, which I promise we will get to, I want to intentionally start our journey into computational content analysis "old school." Given that most content analyses, and academic studies in general, use static datasets, that is, data that we define, collect, and then analyze, we don't need to be worried about some of the problems that big data engineers spend time on. We are not building scalable, flexible systems that must work on new data as the world turns and evolves. We are building a fragile, custom solution to solve specific challenges on known and fixed datasets. This takes the pressure off us in one important way. If it works, and we can verify it works by checking a random sample, then it is good enough. We may analyze the data, write the results, and move on to tenure. Jokes aside, remember, we are fitting a glove, not a one-size-fits-all jacket. This limits our technical challenges, but it also limits what we can say we have solved.

Defining and Justifying "Big Data" in Content Analysis

I am occasionally asked to define what constitutes "big data." My "official" definition, which I offer for the sake of content analysis only, is a dataset that is too large to manually review a representative sample of. This is what makes it "big," in my opinion. After all, computational methods like machine learning or sentiment analysis are only better than humans for two things: scale (Goodfellow et al., 2016), and reliability (Riffe et al., 2014). They invariably are less useful in many other ways. I reason that we should only use computational methods on data when the effort exerted by humans to review a representative would be too great, and therefore we need the scale and reliability that automated technologies bring.

A misconception that has come because of big data analyses in mass communication is the idea that labeling all of the documents in a corpus is necessary to draw conclusions about that corpus. Remember that samples of

DOI: 10.4324/9781003514237-3

data sets are perfectly acceptable if they are drawn appropriately. Determining the appropriate sample size ensures findings are both valid and reliable. Riffe et al. (2014) provide guidance on sampling procedures, suggesting that the traditional standard for content analysis in mass communication often involves examining a sample that is large enough to represent the population, which can be around 10% of the dataset. This percentage is not arbitrary but is based on statistical principles that aim to balance the need for representativeness with practical constraints (Riffe et al., 2014). However, the exact percentage may vary depending on the research question, the variability within the data, and the available resources (Krippendorff, 2018).

With social media data, researchers often must decide among different types of data access. Back when X was Twitter, we had different ways to access content that was on the platform. Different Application Programming Interfaces (APIs) provided different samples of data, there was the firehose (the full stream), the garden hose (a large, but still limited percentage of the stream), or a spritzer stream (a ~1% random sample). Morstatter et al. (2013) discuss the implications of using different sampling strategies and their adequacy for representing the broader social media landscape. They found that the 1% random sample can be significantly different from the firehose, leading to potential biases in the analysis. This suggests that while smaller samples like 1% of a Twitter stream are more manageable, they may not always be sufficient for accurate representation, especially when analyzing rare events or minority opinions (Driscoll & Walker, 2014; Morstatter et al., 2013).

I want to draw your attention to a study I did when I was starting my tenure, "Does Negative Campaign Advertising Stimulate Uncivil Communication on Social Media?" (Hopp & Vargo, 2017). The study explored the relationship between negative political advertising and political incivility on Twitter during the 2012 presidential election. The data for our study was collected from over 140,000 individual Twitter users located in 206 Designated Market Areas. Our data-preparation process was extensive and involved both human and machine labeling. While our dataset was quite large, encompassing over 70 million tweets collected from August 1 to November 6, 2012, only 400,976 of these messages could be resolved to specific Designated Market Areas (DMAs). We wanted to look at the interaction of advertisements in locations, so we needed geo data, and very few users of Twitter opted in. Let this be a lesson that, often, data melts quickly as you segment it to match your desired area of focus.

Determining whether a dataset warrants a "big data analysis" can be boiled down to a simple heuristic: the number of human working hours required. Reviewing a single tweet might take one minute. To label 10% of the dataset, we would need 40,000 minutes—that's about 666 hours. Labeling

a tweet could take mere seconds, but even at a rate of three tweets per second, the labor demanded is excessive and creates an unreasonable expectation for researchers (Kitchin, 2014). Therefore, in these instances, the data is "big" enough to warrant computational content analysis. Under these conditions, my sample of a 400,000-post dataset exemplifies "big data."

However, if I were reviewing video content, the time spent per video would be exponential, and therefore a sample with a much smaller *n*, for instance 10,000 YouTube videos, surely qualifies as big data. 1,000 videos of even just a few minutes in duration each will take a very long time to watch. 10,000 Instagram reels, however, may sound big, but a 10% sample is only 1,000 videos. If they are limited to 30 seconds, it may be reasonable to manually review them in a reasonable amount of time. Generally, the more laborious the content labeling process is, the more justified it is to employ computational methods in content analysis.

Creating a Simple "List of Words" Classifier

Turning back to our study of Twitter, the core of our investigation centered on the classification of tweets based on their incivility level. Faced with an overwhelming volume of microblogging entries (a.k.a., posts), traditional coding methods proved impractical, necessitating an automated approach to draw conclusions. Before we got to Python, we started with an initial qualitative review of a sample of tweets. After reviewing a few hundred tweets, it was obvious to both authors which posts were uncivil. At the time, Twitter was free from moderation, and as a result lots of profanity and insults were hurled at both candidates. We were shocked at the time how much of it was very racially insensitive.

If it were today, I would likely use the generative artificial intelligence approach used later in this book to solve this content analysis problem. It would have offered us the most flexibility and saved me writing my own Python classifier and gathering an exhaustive list of "bad words." That said, generative artificial intelligence requires special computing, is expensive, and ultimately has somewhat of a mind of its own. For these reasons, I'm choosing to start our content-analysis journey using a rules-based classification system. While custom classifiers like this lack the smarts and training/testing functionalities that machine-learning enthusiasts love, they are also simple systems that are easy to understand, and they ultimately afford you the most control over labeling your data.

In addressing the challenge of identifying candidate-specific incivility in tweets, we initiated our research by compiling lists of derogatory and profane terms. These lists, which we sourced from various freely accessible online platforms, included terms that companies such as Google and Yahoo had identified as unsuitable for indexing within search results, reflecting

their inappropriateness for professional settings (Narayanyan, 2018). In the nascent stages of Internet search technology, it was common practice for search engines to maintain databases of words deemed unfit for work-related searches. We merged these publicly available lists with a carefully curated catalog of political candidate names and their associated common misspellings.

This method, often referred to as the "keyword spotting" technique or "bag-of-words" model, provides a straightforward and pragmatic strategy for categorizing textual content by identifying the presence or absence of specific words (Harris, 1954; Salton & McGill, 1983). It's the same solution we talked about in the previous chapter with contextual advertising. Despite its efficacy, this approach presents significant restrictions because it disregards the context and sentiment in which a word is deployed (Pang & Lee, 2008). For instance, our system could misinterpret the use of a potentially offensive term delivered in a humorous or sarcastic tone as objectionably "uncivil." Recognizing this shortcoming we found most uncivil tweets to be quite straightforwardly vulgar.

In fact, over the years authors have asked for our "bad words" lists, and while I have obliged, I've been very clear to reiterate that we created a tailored solution to solve a specific problem with a specific dataset. We did not aim to build an incivility classifier that would even purport to contend with the Jigsaw Perspective API that we talked about in Chapter 2. We printed out a detailed view of the words, the frequency in which they occurred in the corpus (a.k.a., collection, sample), and their context. From that, we created a list to slowly chip away at our problem. We went down the list, reviewing each word one at a time. We reviewed examples of how those words were used in actual posts in the corpus and made a binary decision on whether each word should be considered as a correlate to incivility. Essentially, if a comment contained one or more of these bad words, we labeled it as "uncivil." If it didn't, we labeled it as "civil." This is a very basic form of content analysis, but it was the right-size glove to solve the problem.

Labeling for something you know you're looking for is a somewhat easier problem in content analysis, and information retrieval generally. Pinpointing a concise list of representative keywords can serve as a gateway to identifying positive exemplars of your research interest. As you delve into these instances, they often lead to the discovery of additional pertinent keywords or hashtags. With a greater time investment, you enter a cycle of discovery and refinement until your keyword list encapsulates the essence of the target phenomenon.

This iterative approach underpinned the methodology of our investigation. We continuously expanded our review of tweets, refining our filters until our algorithm consistently identified relevant examples within our dataset. The incremental step of populating our "toxic" lexicon (a.k.a., list)

was crucial for our rudimentary analytic tool. When I'm starting new content analyses today, I still start with this exercise because it can serve four purposes: 1) it compels an in-depth and qualitative understanding of the concept under analysis (Krippendorff, 2018); 2) it yields a preliminary collection of straightforward examples that can enhance a codebook and conceptual definitions; 3) it starts a collection of documents that can be fed to a supervised machine learning training dataset (refer to Chapter 4 for detailed discussion); and 4) for highly specific or simple concepts, such as determining whether a tweet is uncivil toward a political candidate, a well-curated keyword catalog may suffice to address the research question.

Calculating Intercoder Reliability in Python

As I said in the preface, I assume you have some functional working knowledge of Python. If you don't, I strongly recommend using Julius.ai to handle Python code for you.[1] I'm going to assume you understand basic logic and data types. For each chapter, you will find a Python notebook and relevant files that have all the code snippets in one place with comments for you to follow along. I will also include snippets of code and data here in the book for those who want to follow along at a conceptual level. Here is a sample of the kind of data we are working with:

```
typhanieluv | 40.71002717 | -73.78744176 | 0 | "@maddow I'm
echoing Rich Santorum: Mit Romney will be the worst Repub-
lican to run against Obama. So far Soo true!" | 78455847 |
78455847_1343870209275 | false | 0 | false | "Twitter for
Android" | 08/01/2012 21:16:49 | 501 | New York, NY
```

These columns represent various details about each tweet, separated in an "old-school" data format, tab-separated values. The fields are separated by a tab delimiter to prevent this fragile file format from breaking. Inside the fields you see the username of the tweeter ("typhanieluv"), their latitude and longitude, the actual content of the tweet, the source of the tweet ("Twitter for Android"), the timestamp, and the designated market area ("New York, NY"), among other details.

```
# Import the pandas library
import pandas as pd

# Define the path of the file
file_path = 'path/to/your/in_sample.tsv'

# Load the TSV file
# Note: the 'sep' parameter is set to \t, which represents
a tab character in Python, to specify that the file is a TSV
df = pd.read_csv(file_path, sep='\t', header=None)
```

```
# Display the DataFrame
print(df.head())
```

In the above example, replace "path/to/your/in_sample.tsv" with the actual path to your "in_sample.tsv" file.

You may want to add the column names manually using the "names" parameter:

```
column_names = ['user_name', 'latitude', 'longitude',
'content', 'source', 'timestamp', 'designation']
df = pd.read_csv(file_path, sep='\t', header=None,
names=column_names)
```

This will create a DataFrame with the named columns, filled with data from the "in_sample.tsv" file. Once we have our tweets loaded into a DataFrame, we can easily take a random sample of 100 tweets, and add a column for each coder (data annotator)—"Coder A" and "Coder B." Let's save the file, and pretend we went off in the dark for a while and labeled the tweets. Here's a Python code snippet to add additional columns and take a random sample of tweets:

```
# Import necessary libraries
import pandas as pd
import numpy as np

# Define the path of the file
file_path = 'path/to/your/in_sample.tsv'

# Create your column names
column_names = ['user_name', 'latitude', 'longitude',
'num1', 'tweet', vnum2', 'num3', 'bool1', vnum4', 'bool2',
'source',
'timestamp', 'designation', 'location']

# Load the TSV file
df = pd.read_csv(file_path, sep='\t', names=column_names)

# Add new columns for each coder
df['Coder A'] = np.nan
df['Coder B'] = np.nan

# Take a random sample of 100 tweets
df_sample = df.sample(n=100)

# Save the resulting DataFrame to a new TSV file
df_sample.to_csv('path/to/your/sample_tweets.tsv', sep='\t',
index=False)
```

In the above code snippet, replace "`path/to/your/in_sample.tsv`" with the actual path to your "`in_sample.tsv`" file. Similarly, replace "`path/to/your/sample_tweets.tsv`" with the path where you want to store your output sample file.

When you're doing your content analysis, you'll want to label your data using a spreadsheet program like Microsoft Excel, Apple Numbers, or Google Sheets. Create columns like the "Coder A" and "Coder B" columns in the saved "`sample_tweets.tsv`" file. This procedure assumes that two coders ("Coder A" and "Coder B") are manually labeling the sampled tweets.

Once your manual labeling process is finished, you can load the sample data back into Python using "`pd.read_csv('path/to/your/sample_tweets.tsv', sep='\t')`" for the next phase of your analysis.

We're going to make sure that the two coders (Coder A = our script; Coder B = me) agree on what incivility is. Let's calculate intercoder reliability to verify that the coders have an acceptable agreement (Lombard et al., 2002). Intercoder reliability is a critical metric in content analysis, representing the degree of alignment among different coders who classify or label dataset contents (Neuendorf, 2002). It essentially assesses whether our codebook and annotators (human or android) consistently identify and interpret data (Krippendorff, 2018). To quantify this reliability, several statistical measures can be employed, such as percent agreement, Cohen's Kappa (Cohen, 1960), Krippendorff's Alpha (Krippendorff, 2004), and Fleiss' Kappa (Fleiss et al., 2003).

As you calculate the proportion of decisions in which coders concur, percent agreement emerges as the most rudimentary metric. Its major limitation is that it doesn't factor in any concurrence that might arise by sheer coincidence. Cohen's Kappa (κ), building upon the foundation of percent agreement, corrects for chance agreement. This index spans from –1 to +1, with +1 denoting complete concordance, zero representing an agreement level no better than randomness, and –1 signaling absolute discord. Cohen's Kappa becomes particularly relevant when dealing with just two raters, each evaluating every item (McHugh, 2012).

When your analysis involves more than a pair of raters, or when not all items are evaluated by every rater, you may turn to Krippendorff's Alpha. This broader statistical construct quantifies agreement across various categories and accommodates data of multiple types, such as nominal, ordinal, or interval. Fleiss' Kappa extends the utility of Cohen's Kappa and is applicable when numerous raters categorize a collection of items or classes.

In a social science investigation like the one outlined, with two individuals ("Coder A" and "Coder B") manually categorizing tweets as "uncivil" or "not uncivil," Cohen's kappa would aptly assist in appraising the intercoder reliability. Computing Cohen's kappa involves first determining the observed agreement rate—the extent to which the coders' decisions concur—and then gauging the expected agreement rate—the likelihood of coincidental coder

concurrence. You arrive at the kappa statistic by deducting the expected agreement from the observed agreement and dividing the resultant value by the difference between one and the expected agreement (Cohen, 1960).

Here is a Python function to gauge intercoder reliability:

```
def intercoder_reliability(df, coder1, coder2):
coder1_labels = df[coder1]
coder2_labels = df[coder2]

# Calculate observed agreement
observed_agreement = sum(coder1_labels == coder2_labels) /
len(coder1_labels)

# Calculate expected agreement
coder1_incivility_rate = sum(coder1_labels == 1) /
len(coder1_labels)
coder2_incivility_rate = sum(coder2_labels == 1) /
len(coder2_labels)
coder1_civility_rate = sum(coder1_labels == 0) /
len(coder1_labels)
coder2_civility_rate = sum(coder2_labels == 0) /
len(coder2_labels)

expected_agreement = (coder1_incivility_rate *
coder2_incivility_rate) + (coder1_civility_rate *
coder2_civility_rate)

# Calculate kappa
kappa = (observed_agreement - expected_agreement) / (1 -
expected_agreement)

return kappa
```

Please note that for more complex annotation tasks, more elaborate intercoder reliability metrics such as Krippendorf's Alpha or Fleiss' Kappa, which allow for multiple coders and the possibility of missing data, may be more appropriate. This simple calculation should work for the binary classification task described in the text. Now, let's load our data and calculate intercoder reliability. To load the "`intercoder_reliability.xls`" file and call the "`intercoder_reliability`" function on the "Coder A" and "Coder B" columns, you would first import the necessary libraries (pandas in this case) and then use the "`pd.read_excel()`" function to load the spreadsheet data into a pandas DataFrame.

Afterward, you would invoke the "`intercoder_reliability()`" function, passing the loaded DataFrame and the column names "Coder A" and "Coder B" as arguments.

Here's a Python code snippet illustrating how to do it:

```
# Import the pandas library
import pandas as pd

# Define the path of the file
file_path = 'path/to/your/intercoder_reliability.xlsx'

# Load the Excel file
df = pd.read_excel(file_path)
# Run the 'intercoder_reliability' function on 'Coder A'
and 'Coder B' columns
kappa = intercoder_reliability(df, 'Coder A', 'Coder B')

# Print the calculated Cohen's kappa
print("Cohen's Kappa is", kappa)
```

Replace "`path/to/your/intercoder_reliability.xlsx`" with the actual path to the file. This script will output the calculated Cohen's kappa, which is the measure of intercoder reliability in this case. The higher the value of kappa, the greater the level of agreement between the two coders. A negative kappa indicates a level of agreement that is worse than random.

You may be wondering about the benchmarks for a satisfactory Cohen's kappa coefficient. The answer is nuanced and contingent on the complexity of the problem at hand. In situations comparable to ours, which are more straightforward, a kappa value exceeding 0.8 is typically anticipated. For issues that are less clear or require more profound understanding, the expected kappa may be markedly lower. While there is no universal criterion for Cohen's kappa, as a guideline, values above 0.8 are often regarded as "almost perfect" agreement. Scores ranging from 0.6 to 0.79 indicate "substantial" agreement, 0.4 to 0.59 denote "moderate" agreement, 0.2 to 0.39 reflect "fair" agreement, 0 to 0.19 are considered "slight" agreement, and values below 0 signify "poor" agreement (Landis & Koch, 1977).

Building a Simple Classifier in Python

Now that we've taken some time to validate that our human coders are labeling data as we'd expect, let's take some time to unpack how I've instructed the computer to label data and discuss a little natural language processing. We don't need to know a ton to solve this problem—just a few concepts are going to take us a long way. If you refer to the "Text Preprocessing and Tokenization" section of the Python notebook included in this chapter, you can see the related code. I won't include it here for space considerations.

The "`tokenize(text)`" function breaks down the text into individual words or "tokens." This is done using regular expressions, which are patterns used to match character combinations in strings. In this case, the regular expressions "`WORD_PAT`" and "`WORD_PAIR`" are used to match individual words and pairs of words, respectively (Bird et al., 2009).

The function starts by initializing an empty list of tokens. It then enters a loop that continues until the end of the text. In each iteration of the loop, the function first tries to find a match for a word pair in the remaining text. If it finds a match, it checks if the pair (stripped of any leading or trailing punctuation and converted to lowercase) is in the list of incivility words. If it is, the pair is added to the list of tokens. If it's not, the function tries to find a match for a single word in the remaining text.

Because the word "hate" is included in the incivility words CSV file, the "`tokenize(text)`" function would recognize it as an incivility word and add it to the list of tokens. However, the function as it is currently written would not recognize "hater" or "haters" as matches for "hate." This is because the function is looking for exact matches: it checks if a token (a word or word pair from the tweet text) is in the list of incivility words, not if an incivility word is in a token.

This is a limitation of the current approach. One way to address this would be to use stemming or lemmatization, which are techniques for reducing words to their root form. For example, the stem of "hater" and "haters" is "hate." If we applied stemming to the tokens before checking them against the list of incivility words, then "hater" and "haters" would match "hate." To incorporate lemmatization into the function, we would need to use a library like Natural Language Toolkit (NLTK), which provides a lemmatizer that reduces words to their base or root form (lemma). In the Python notebook for this chapter, I've prepared a version of the code that uses lemmatization at the end of the "`tokenize(text)`" function. I've "commented it out" by inserting #'s in front of the code to ensure it doesn't run. To run this version, uncomment it out (CMD +/on Mac in Google Colab, my preferred way to open Python notebooks), and comment out the version of code that does not use lemmatization (again, CMD +/on Mac).

This revised function will now lemmatize each word or word pair before checking if it's in the list of incivility words. This means that variations of a word, like "hater" or "haters," will be reduced to their root form "hate" and correctly identified as an incivility word if "hate" is in the list. Beware, however; we didn't end up doing this in our study, because it increased the occurrence of false positives. For example, the word "hat" would also be reduced to the stem "hat," which would match "hate" if we use lemmatization. So, if we use a lemmatized approach, we need to be extremely stringent in our uses of lemmas, as they are bound to have more unintended matches than our exact matching approach.

The "`get_word_scores()`" function reads the list of incivility words and their scores from a file. The score of a word indicates its degree of incivility. For example, "hate" might have a higher score than "dislike." While this is a feature we did not ultimately use in our paper, this general functionality could allow you to score some words as more severely uncivil than other words.

The "`remove_at_mentions(text)`" function is used to preprocess the text of the tweet by removing any @mentions. This is done using regular expressions, a powerful tool for matching and manipulating strings. In this case, the regular expression "@\.?\w+" matches any word that starts with "@" which is how mentions are represented on Twitter. For example, the tweet "@maddow I love your show!" would be transformed into "I love your show!".

The "`preprocess(tweet)`" function is a wrapper function that calls "`remove_at_mentions(text)`." This function could be expanded in the future to include other preprocessing steps, such as removing URLs or hashtags, converting the text to lowercase, or removing punctuation.

The "`main()`" function is where the action happens. It reads the tweets from a file, preprocesses them, tokenizes them, and calculates their incivility score. The incivility score of a tweet is the sum of the scores of its incivility words. The function also boosts the score of a tweet if it contains words with exclamation marks or capital letters, as these are often used to express strong emotions. For example, the tweet "I HATE Python!" would have a higher score than "I hate Python."

Finally, the script prints the incivility score of each tweet and stores that score. In all we created a tool that had a one-time use, to classify 400,000 tweets. We kept tweaking and editing our Python script, each time reviewing small samples of its output labels, until we felt that the machine was accurately labeling the data. At the top of this chapter, we verified that this was indeed the case by calculating intercoder reliability, where Coder A was the Python script. In all, the script identified over 650 words associated with candidate-relevant incivility. The time spent here was primarily on reading 800+ tweets to generate the word lists most associated with incivility and also tweaking the logic of the code to produce the correct labels. Most importantly, we checked if any uncivil words were not detected by the computer.

The external validity percentages were over 98%, indicating high precision. This means that 98% of the data labeled was exactly how we expected it to be. Why? Well, first, incivility in this dataset was blatant and easy for humans to spot. We knew what we were looking for, and it was easy to see it manifested in specific, unique words. We then tailored our tool to a specific use, improved it iteratively, and our dataset remained static during

that time. Interestingly, and perhaps not truly necessary (read: the reviewers made us do it), we also ran several additional external validity checks. We reasoned that if our "algorithm" was truly measuring incivility, it would also negatively correlate with sentiment, as most uncivil content is negative in valence. Additionally, we suspected a positive correlation with psychological (and not sexual) arousal, given that incivility is a highly arousing set of emotional responses. To do this, we used open-source lexicon classifiers that measured these concepts and labeled the data with them. As we expected, incivility was negatively correlated with sentiment and positively correlated with arousal, even further suggesting that we were approximately measuring the concept of interest.

After text classification was used to derive incivility scores for each tweet, we then created incivility scores that were then averaged for each Twitter user, and these average scores were aggregated at the Designated Marketing Area (DMA) level to create an overall incivility score for each DMA. When compared against advertising DMA data, we now had a measure of candidate-directed incivility as exhibited through Twitter that was directly comparable. The linking of DMA advertising spend data with DMA-specific data from Twitter provided a nuanced view of public sentiment, effectively tracking how advertising spending impacted public discourse.

In our example above, content analysis was the vehicle that drove the integration of different datasets (Krippendorff, 2018). By applying content analysis methods to Twitter data, we were able to label each tweet with an incivility score. The raw tweets, essentially unstructured data, were thus transformed into structured data that could be compared and matched with the structured DMA advertising spend data. When two structured datasets are married like this, the result is a rich, multi-layered dataset that can be mined for invaluable insights. For instance, the analyses that could be performed on such a dataset range from looking at correlations between advertising spend and public discourse, comparing public sentiment across different DMAs to measuring the impact of local news events on public sentiment. The possibilities are wide-ranging and ever-evolving as new methods and techniques of content analysis are developed.

The promise of marrying data extends beyond social media and advertising spend data. Labeled media data, in general, could be coupled with a vast array of other datasets. For instance, census data, socioeconomic indicators, weather data, or even data from satellite imagery could be combined with labeled media data to examine a variety of phenomena—from studying correlations between media coverage and changes in demographic trends to measuring the impact of environmental conditions on media discourse. As we continue to generate and accumulate unprecedented amounts of data in our digital age, these methods will become more critical in making sense

of our complex world. They will enable us to uncover patterns and associations that would otherwise remain hidden, providing valuable insights into human behavior and society.

In closing this chapter, I hope you found that the accompanying code is a very simple but effective way to measure a concept that is manifested in text. You can easily repurpose this tool by simply changing the words that it looks for. For instance, you could look for issues, attributes, frames, or any other straightforward concept. It illustrates several key concepts in natural language processing, including tokenization, regular expressions, and text classification. However, such a simple approach has its limitations. For instance, it does not account for the context in which words are used, or for figurative language, such as sarcasm or irony. More sophisticated measurements would require the use of supervised machine learning, a method that involves feeding the algorithm labeled examples from which it can learn, or Generative AI, an approach involving teaching the machine model to learn the characteristics of a given category and then generate new, original content that fits into the same category. We will cover these topics directly in the chapters to come.

Note

1 Give it a try for free by visiting: `https://julius.ai`. Just upload the file and, in your own words, tell it to load the data into a pandas dataframe. For subsequent steps, describe to the AI what you want to do, and with a little trial and error, it will do almost everything we cover in this book, with the exception of generative AI.

References

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python: Analyzing text with the Natural Language Toolkit*. O'Reilly Media, Inc.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, *20*(1), 37–46.

Driscoll, K., & Walker, S. (2014). Big data, big questions| Working within a black box: Transparency in the collection and production of big Twitter data. *International Journal of Communication*, *8*(2014), 1745–1764.

Fleiss, J. L., Levin, B., & Paik, M. C. (2003). *Statistical methods for rates and proportions* (3rd ed.). John Wiley & Sons.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Harris, Z. (1954). Distributional structure. *Word*, *10*(2–3), 146–162.

Hopp, T., & Vargo, C. J. (2017). Does negative campaign advertising stimulate uncivil communication on social media? Measuring audience response using big data. *Computers in Human Behavior*, *68*, 368–377. https://doi.org/10.1016/j.chb.2016.11.034

Kitchin, R. (2014). Big Data, new epistemologies and paradigm shifts. *Big Data & Society*, *1*(1). https://doi.org/10.1177/2053951714528481

Krippendorff, K. (2004). *Content analysis: An introduction to its methodology* (2nd ed.). Sage.

Krippendorff, K. (2018). *Content analysis: An introduction to its methodology* (4th ed.). Sage.

Landis, J. R., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, *33*(1), 159–174. https://doi.org/10.2307/2529310

Lombard, M., Snyder-Duch, J., & Bracken, C. C. (2002). Content analysis in mass communication: Assessment and reporting of intercoder reliability. *Human Communication Research*, *28*(4), 587–604. https://doi.org/10.1111/j.1468-2958.2002.tb00826.x

McHugh, M. L. (2012). Interrater reliability: The kappa statistic. *Biochemia Medica*, *22*(3), 276–282. https://doi.org/10.11613/BM.2012.031

Morstatter, F., Pfeffer, J., Liu, H., & Carley, K. M. (2013). Is the sample good enough? Comparing data from Twitter's Streaming API with Twitter's Firehose. *Proceedings of the Seventh International AAAI Conference on Weblogs and Social Media*, *1306*(1), 400–408. http://doi.org/10.1609/icwsm.v7i1.14401.

Narayanan, B. K. (2018). Adult content filtering: Restricting minor audience from accessing inappropriate internet content. *Education and Information Technologies*, *23*(4), 2719–2735. https://doi.org/10.1007/s10639-018-9738-y

Neuendorf, K. A. (2002). *The content analysis guidebook*. Sage.

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, *2*(1–2), 1–135. https://doi.org/10.1561/1500000011

Riffe, D., Lacy, S., & Fico, F. (2014). *Analyzing media messages: Using quantitative content analysis in research* (3rd ed.). Routledge.

Salton, G., & McGill, M. J. (1983). *Introduction to modern information retrieval*. McGraw-Hill.

Chapter 4

Supervised Machine Learning with BERT for Content Analysis

The previous chapter focused on a basic keyword spotting approach for identifying incivility in tweets surrounding the 2012 election. It was simple, but adding words to a list as we read more tweets eventually built a tool to do the job. As this book goes on, we'll dial up the complexity of the method on how to solve problems like this. In this next approach, we'll lose the ability to add words to a list, but also be freed of that tedious responsibility, and instead just be asked to label documents like we would normally in a content analysis.

I want to draw your attention to another study, this time one where I used *supervised machine learning* to classify documents. The study (Tomeny et al., 2017) stemmed from a conversation I had with a colleague who was a psychology professor studying autism at the University of Alabama. He wanted to know *why* people shared a myth that persists to this day, that vaccines cause autism. He wanted to know why the myth persisted on social media. At the time, and sadly no longer the case, I had good access to Twitter data, and I found 549,972 tweets that mentioned vaccines and autism. The big question we needed to answer was the position the tweet was taking. Did the tweet support the idea that "vaccines cause autism," or did the tweet refute that claim? Using a machine-learning algorithm we analyzed the tweets and separated which contained anti-vaccine sentiments from the ones that rebuffed the idea. The data revealed that anti-vaccine tweet volume increased following vaccine-related news coverage and was geographically concentrated in certain areas. Certain factors such as the average household income of an area positively correlated to higher percentages of anti-vax sentiment.

Why not stick to keyword spotting? While the lexicon approach served well in our prior focus on political incivility, it has limitations. It's heavily reliant on a predefined list of words and can misclassify tweets due to the context in which words are used; this is an inherent complexity of human languages. Unlike incivility, which has been studied and distilled down to "bad

DOI: 10.4324/9781003514237-4

word" lists, exactly what constitutes an anti-vax tweet is context-dependent. It's hard to argue that a tweet that has the "f" word is civil, even if it has positive sentiment. However, when you imagine the almost infinite ways people could show their support for or against an idea, it becomes less immediately clear what signals we will need to latch on to classify the tweets.

The literature suggested we search for keywords like #CDCWhistleblower where anti-vaccine people band together to discuss vaccines and autism. However, those keywords did not just surface negative discussion, but also people debunking and trying to dissuade people that the link was real. Even keywords like "mercury" or "delayed schedule" are not dead ringers because people can express support for or against any divisive issue. As such, we turned to supervised machine learning (Kotsiantis, 2007). Supervised learning is a type of machine learning where the model is "trained" using labeled data (Alpaydin, 2020). In content analysis, you can think of words as coefficients in a regression analysis; each brings a certain weight or importance that contributes to the output or prediction (James et al., 2021). By "learning" from the existing data, the model fits its best guess as to the importance of each word/coefficient. It learns the optimal fit by trying to recreate the labels found in the training data as closely as possible. This flexibility is in some ways amazing. We no longer must explicitly define words to make predictions. We can make predictions or decisions without explicitly programming them, but instead by allowing our labels to give the computer the evidence it needs to make the classification.

Preparing Data for a Supervised Machine-Learning Algorithm

First, of course, we had to define the concept. To train the algorithm, two researchers coded 3,371 tweets into two categories: (1) anti-vaccine and (2) all other tweets. We instructed our coders to assign the anti-vaccine label when vaccines were portrayed as dangerous, ineffective, or negative and mentioned a potential causal link to ASD (e.g., "CDC whistleblower confesses to publishing fraudulent data to obfuscate link between vaccines and autism," "RT @AnonCorpWatch: Autism is mainly caused by mercury present in vaccines"). The coders made their judgments independently and had perfect intercoder reliability due to the straightforward nature of anti-vaccination tweets. You can take a look at the training data for the project by simply loading it as a DataFrame in Python, or the old-fashioned way by opening it in Excel. Either way, you'll see that the first column to the left is the text of tweets, and then a simple "0" or "2" in the annotation column, telling the computer if the tweet is anti-vax (2) or not (0). We then allowed a supervised machine-learning algorithm to learn the relationship

between the words used in the tweets (which we consider as features) and the labels we provide.

Let's imagine our algorithm assigns a coefficient of –0.3 to the word "harmful," +0.4 to "mercury," and +0.5 to "#CDCWhistleblower." In simple terms, these coefficients represent the "weight" of each word in determining whether a tweet is likely to be pro-vax or anti-vax. The higher the coefficient (positive or negative), the more predictive the word tends to be. Positive coefficients mean that a particular word is predictively associated with anti-vax sentiment, whereas negative coefficients are associated with pro-vaccine sentiment. If we look at the word "harmful," for instance, the negative coefficient suggests that this word is often used in the context of pro-vaccine discussions, perhaps in sentences discussing the harmful effects of diseases that vaccines prevent. Conversely, the positive coefficients for "mercury" and "#CDCWhistleblower" suggest that these terms are more distinctively associated with the anti-vax sentiment.

In this way, the machine-learning algorithm is analyzing and weighting the importance of these words based on the relationships it learns from the training data. The more times "mercury" is mentioned in anti-vax tweets, the stronger its coefficient will become. For this project, we used a simple bag-of-words (BoW) model (Kotsiantis, 2007). Bag-of-words models extract all words in each document and look at their co-occurrences with classes, here not anti-vax, or anti-vax. The journey begins with a corpus of documents, a.k.a., a large and structured set of texts. In our study, the corpus consists of all the tweets we want to analyze. Each tweet in the corpus is referred to as a document.

In the initial phase of the process, the corpus undergoes pre-processing. This step usually involves the conversion of all text to lowercase to ensure uniformity (Manning et al., 2008), unless there are specific reasons why the capitalization would help us understand the context and usage of words. It also includes tokenization, the process in which a document is broken up into individual words or tokens (Baeza-Yates & Ribeiro-Neto, 2011). In addition, words that offer little unique meaning, (a.k.a., stop words) are typically removed, including words like "the," "and," "is" (Han et al., 2011). Depending on the specifics of a project, this step may also include lemmatization or stemming, strategies to reduce multiple forms of a word down to its base form (for example, "running," "runs," "ran" all to "run"), and the removal of special characters or numbers (Manning et al., 2008).

Once pre-processing is complete, the BoW model converts the corpus into a document-term matrix (Bishop, 2006). In this matrix, the rows represent the documents (in this case, individual tweets), and the columns correspond to the unique terms or words obtained from all the documents. Each cell in the matrix represents the frequency of a term in a particular document (Hastie et al., 2009).

Let's consider the following tweets:

"Vaccines cause autism! #CDCWhistleblower"
"Mercury in vaccines is harmful"
"Vaccines are safe and effective"
"Autism is not caused by vaccines"
"Vaccines save lives, don't believe the myths"

Here's the corresponding document-term matrix:

	and	*are*	*aut-ism*	*believe*	*by*	*cause*	*caused*	*cdcwhistle-blower*	*don*	*effe-ctive*	*harm-ful*	*in*	*is*	*lives*	*mercury*	*myths*	*not*	*safe*	*save*	*the*	*vaccines*
0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	1
2	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1
3	0	0	1	0	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	1
4	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	0	0	1	1	1

In a BoW representation, the information about the order or structure of words in the document is discarded. In effect, the model treats each document as just a collection of words. We'll address these major limitations with deep learning in the next chapter (Bishop, 2006). By ignoring the context in which a word is used and the order in which words are presented in a story, we're ultimately losing some information about the document. It's a trade-off we accept for simple problems in order to represent our data in a document-term matrix.

This document-term matrix serves as an input feature set for the learning algorithm (Hastie et al., 2009). The columns correspond to the "words" we referred to in the initial article, and each word's "weight" or importance is the frequency of its appearance in the document. Could you simply fit a multiple regression where the predictor (X) variables are the text columns, and Y is the label of the tweet (anti-vax/vax)? Absolutely! That's essentially what we're doing with the text processing, getting down to one very wide tabular dataset. In reality, we'll have way too many predictors to fit into a simple multiple regression. Luckily, there are hundreds of models that were designed to handle issues with many variables, like multicollinearity (Bishop, 2006). Once we've gotten our text data represented as a document-term matrix, it becomes relatively simple to workshop these models, as we'll see in the code later in this chapter.

Building a Supervised Machine-Learning Algorithm

The resulting matrix is often high-dimensional, containing a column for practically every unique word in the corpus, which may number in the thousands or millions. In this way, through the help of a BoW classifier, we were able to turn a collection of texts into a structured numerical format suitable for machine learning—vectors (Han et al., 2011). The vectors we end up with represent a simplified snapshot of our corpus, reducing complex text to a list of numbers, ready to be fed into our machine-learning algorithm (Hastie et al., 2009). Each vector represents a unique document, with each word in the document being a feature in that vector. As stated earlier, the weight or importance of the feature comes from the frequency of its appearance in the document (Bishop, 2006). Essentially, each vector is a numerical representation of the contents of a document.

The machine-learning algorithm intuitively maps the vectors (documents) with the labels or outcomes (in our case, "not anti-vax" or "anti-vax"). The coefficients assigned to each term or keyword, as exemplified in the article by a coefficient of –0.3 to "harmful," +0.4 to "mercury," and +0.5 to "#CDCWhistleblower," come into play here. To sum scores for a document, the machine-learning algorithm multiplies each word's frequency by its corresponding learned coefficient and then takes a sum of all scores (Hastie et al., 2009).

For instance, if the word "harmful" appears three times in a tweet, its total contribution to the score is –0.3 (its coefficient) * 3 (frequency) = –0.9. Similarly, the total contribution of the word "mercury" might be +0.4 * 1 = +0.4 if it appears once in the tweet. The score for the tweet is the sum of these contributions from each word in the tweet (Bishop, 2006).

The summed score for each document (tweet) is then used to predict the label or classification category. In our context, if the total score for a given tweet is positive, it could be predicted as "anti-vax." Conversely, tweets with negative scores might be predicted as "not anti-vax." This approach allows us to use the frequency of individual words and their corresponding coefficients to understand the underlying sentiment within the text based on the scores computed, hence making an informed decision or prediction (Hastie et al., 2009).

The actual specifics of how scoring is done and how predictions are made can vary based on the specific machine-learning algorithm being used. However, the general principle of using summed scores to make predictions holds across different methods. In this study, we used a bag-of-words classifier through the machine-learning workbench, LightSide, an open-source platform that performs feature extraction. We also used a model called LibLinear (Fan et al., 2008). LibLinear is a machine-learning tool developed by National Taiwan University to solve large-scale linear classification problems. It was born out of a desire to efficiently handle high dimensional data,

such as text, where the number of features can be in the order of millions. Until its inception, the large size and high dimensionality of such datasets proved a challenge for existing techniques, both in terms of computational efficiency and the quality of solutions (Fan et al., 2008).

LibLinear employs a coordinate descent method that can be used for solving linear Support Vector Machines (SVMs) and logistic regression models (Fan et al., 2008). SVM is a commonly used machine-learning model for classification and regression problems. In our study, the SVM solves the challenge of high-dimensionality (e.g., lots of words) and large volumes of text data while classifying tweets (Bishop, 2006). Similar models to LibLinear that are still being used today include LIBSVM, SVMlight, and SVMLIB. These libraries also offer tools for large-scale support vector machine (SVM) regression and classification, but they tend to be more suited to smaller-scale problems where the number of features is less than the number of instances, contrasting with the strength of LibLinear in handling high dimensional data (Fan et al., 2008).[1]

Let's consider an example of text classification in Python using LinearSVC from Scikit-learn. Remember, you can find the entire code for this chapter in the corresponding Python notebook. Look at how few lines of code are needed to do supervised machine learning:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split

# Let's assume we have some data
X = ['vaccines cause autism', 'vaccines don't cause au-
tism'] # your text data
y = [1,0] # your labels

# Splitting data into training and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Making a pipeline that will first turn our raw text into a
Bag of Words (BoW)
# and then runs a Linear SVC on it
text_clf = Pipeline([('vect', CountVectorizer()), ('clf',
LinearSVC())])

# Training the classifier
text_clf.fit(X_train, y_train)

# Testing the classifier
predicted = text_clf.predict(X_test)
```

In this example, the "`vect`" stage of the pipeline performs the feature extraction i.e., it turns the raw text into a document-term matrix similar to the BoW method used in LibLinear. This is followed by the "`clf`" stage that trains the LinearSVC model with the training data. Finally, predictions are made on the test data. It really can be this little Python code, as Scikit-learn does most of the heavy lifting for us.

Evaluating a Supervised Machine-Learning Algorithm

Once we build a machine-learning algorithm, we must evaluate it. When classifying anti-vax tweets using a supervised machine-learning algorithm, we're essentially trying to create a model that can accurately and reliably identify patterns that correspond to anti-vaccine sentiments. When we talk about evaluating our model, we're concerned with two main things: How well does our model perform, and can we trust its predictions? This is where cross-validation (CV; Kohavi, 1995) and metrics like accuracy and Kappa (McHugh, 2012) come into play.

Cross-validation is a technique that helps us ensure that our model's performance is not just a fluke of the sample of tweets we used to test its performance (Kohavi, 1995). By systematically splitting our data into different training and testing sets, we can test how well our model generalizes to new, unseen data. This is akin to testing a theory in different contexts to ensure it holds up and isn't just a product of specific circumstances. Without this protection, it could be just due to chance that our test set is filled with very blatant, easy-to-label data, thereby artificially exaggerating the true ability of our algorithm, or vice versa with difficult examples.

Accuracy is a straightforward measure that tells us what proportion of the time our model is getting things right. In the context of anti-vax tweets, a high accuracy would mean that our model is correctly identifying a high percentage of tweets as either anti-vax or not anti-vax. However, accuracy doesn't tell the whole story, especially if our data is unbalanced (Fernández-Delgado et al., 2014).

Though accuracy seems like a straightforward and intuitive measure, it can be misleading in cases where the dataset is imbalanced. For instance, say we have 95 not anti-vax and five anti-vax tweets. Even a naive model that predicts all tweets as not anti-vax would have an accuracy of 95%! It would only be wrong five times out of 100, but it would totally miss the concept of interest, here anti-vax tweets. To cope with such limitations, accuracy is often used in conjunction with other metrics such as precision, recall, F1-score, and Kappa, which provide a more balanced evaluation (Powers, 2011).

Kappa is particularly useful because it accounts for the accuracy that would occur by random chance (McHugh, 2012). This is important in social science research because we want to be confident that our findings are

not due to random variation but reflect actual patterns in the data. A high Kappa score means that the agreement between our model's classifications and the true labels is significantly better than what would be expected just by guessing. This is why we use it to calculate the agreement between two human coders, and it's equally valid to use it here as well.

For example, let's say we have a dataset of 1000 tweets with 100 of them being anti-vax. If the model predicts 90 of the anti-vax tweets correctly but also incorrectly labels 200 pro-vax tweets as anti-vax, the accuracy would be (700 correct pro-vax + 90 correct anti-vax) / 1000 = 79%. However, this doesn't reflect the poor performance of the model on the majority class (pro-vax tweets). The Kappa score would reveal this since it would be much lower than the accuracy due to the high number of false positives (McHugh, 2012). There are 1000–100 = 900 pro-vax tweets. The model labeled 200 of them as anti-vax so the model would have labeled 700 correctly. The poor performance is indeed more pronounced in the majority class, with a correct classification rate of 700/900 = 77.78%, compared to 90/100 = 90% for the minority class.

Now, let's try to build a LinearSVC model with the data. This model is similar to the one that I used to build the analysis on and is generally considered to be a better choice for big datasets of text. First, let's import the necessary libraries:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline
from sklearn import metrics
```

We can start by loading the dataset. Considering the article text above, the data has been coded with two labels: (1) anti-vax and (2) not anti-vax. For simplicity's sake, we will use a basic dataset instead of loading one from an external source. Let's assume that you have the "`forml.xlsx`" file in your current working directory. Here's a basic way to load an Excel file into Python using the pandas library:

```
import pandas as pd

# Load the Excel file
df = pd.read_excel('forml.xlsx')

# Display the first 5 rows of the dataframe
print(df.head())
```

In this code snippet, the pandas function "`read_excel()`" is used to load the "`forml.xlsx`" file into a DataFrame object (df).[2] A DataFrame is a two-dimensional labeled data structure with columns of potentially different types, similar to a spreadsheet or SQL table. The "`head()`" function is then used to print out the first five rows of the DataFrame to the console, allowing you to get a quick glance at your data.

Next, we split our data into training and testing sets. This is done using "`train_test_split`" from the `sklearn.model_selection` library. We will use 80% of the data for training and 20% for testing. Remember, the model learns from the trends, patterns, and relationships in the training data. That training data is labeled and came from our human content analysis. It has both the input features (in this case, the text of the tweets) and the corresponding output (the classification labels of "anti-vax" or "not anti-vax"). This allows the model to form an understanding or mapping from the input features to the output label.

The training phase is iterative, with the model repeatedly making predictions on the training data and adjusting its internal parameters depending on how accurate its predictions are with the help of a learning algorithm (James et al., 2021). Over time, the model adjusts itself to minimize the discrepancy, or the difference, between its predictions and the actual reality, thereby improving its prediction accuracy.

Once the model has been trained, it must be tested to evaluate its generalizability to unseen data. This is where the testing set, the remaining 20% of the data, comes into play (Kuhn & Johnson, 2019). The testing set is data that the model did not encounter during its training phase, and it's used to evaluate how well the model can generalize its learning to new, unseen instances.

Holding back 20% of the data as the test set is how we know if this process actually worked and the computer actually learned.[3] By testing on unseen data, we ensure an unbiased measurement of the model's true performance and its capability to generalize learning (Provost & Fawcett, 2013). If we only measure its performance on the training data, we risk over-optimism in the model's predictive power due to what is known as overfitting, where the model learns the training data too well, to the point where it is not able to generalize well to unseen data (Goodfellow et al., 2016). An overfit model becomes so well-adjusted to the training data that it fails to generalize well to new, unseen data. Overfit models essentially memorize the training data and perform poorly when confronted with new situations that they haven't seen during training. It reacts to the "noise" and randomness in the training data, mistaking these for useful information.

While both social science research and machine learning aim to decipher patterns within complex data and make meaningful inferences, the methodologies they typically employ have some key differences, particularly

when it comes to the use (or non-use) of hold-out data for evaluation (Provost & Fawcett, 2013). In traditional social science research, hypothesis testing is often the objective. Models are fit on all available data to estimate the parameters of interest (such as coefficients in a regression model), and the statistical significance of these parameters is used as an indication of the presence or absence of certain effects. In this paradigm, the primary concern is developing a comprehensive understanding of the relationships within the data at hand and statistically justifying these relationships. The focus is less on the model's ability to generalize its findings to unseen data. Consequently, there is usually no practice for setting aside a portion of the data to test the model's predictive accuracy.

Conversely, the core aim of most machine-learning tasks is predictive accuracy. Given this objective, it is imperative to assess how a trained model performs on unseen data. This concern is addressed by the practice of holding out a subset of the training data as a "test set" (James et al., 2021).

Consider a class of students studying for an exam. Throughout the semester, they go through several topics, completing different exercises and practice problems related to those topics—let's say this is the "training data." When studying for the exam, a wise strategy would be to understand the underlying principles and methods taught in class, so that they can apply them to any problem, be it one they have seen before or a new one. The students who do this are like a well-trained machine-learning model—they've learned from the training data, but they're also able to generalize their knowledge to unseen problems on the exam (Kuhn & Johnson, 2019).

However, suppose some students opt for memorization. They memorize every single practice problem seen throughout the semester, without trying to understand the principles behind them. They are banking on the hope that the exam questions will be the same as the training data. These students are taking the risk of "overfitting" their study: they are perfect at reproducing the problems they've seen but may fail when faced with new, unseen problems (Goodfellow et al., 2016).

The actual exam, in this case, can be likened to the "test set." It evaluates how well students—or, in the machine-learning context, models—can apply what they've learned to new, unseen problems. By including questions that students have not directly seen before, the teacher ensures that students' understanding is not narrowly fitted to the practice problems given throughout the semester, thereby avoiding overfitting and encouraging genuine understanding.

```
# Splitting data into training and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Next, we use sklearn's `TfidfVectorizer` to convert our text data into numerical form. We then use those text features to train a `LinearSVC` classifier.

```
# Making a pipeline that will first turn our raw text into a
Bag of Words (BoW)
# and then runs a Linear SVC on it
text_clf = Pipeline([('tfidf', TfidfVectorizer()), ('clf',
LinearSVC())])
```

After creating our pipeline, we are then going to use the fit function to train our classifier.

```
# Training the classifier
text_clf.fit(X_train, y_train)
```

Now that our classifier has been trained, we can use the predict function to classify our test data.

```
# Testing the classifier
predicted = text_clf.predict(X_test)
```

Finally, we measure the performance of our algorithm by checking its accuracy with metrics from `sklearn`. This is the model's performance on data it's seen, so we should hope for this value to be high if it's a good "memorizer," as we said in our student and test-taking example earlier.

```
# Calculating and printing accuracy
accuracy = metrics.accuracy_score(y_test, predicted)
# Converting it to a percentage and rounding it to 2 deci-
mal places
accuracy_percent = round(accuracy * 100, 2)
print(f'Accuracy: The "accuracy_percent" variable repre-
sents the accuracy of our machine learning model imple-
mented using scikit-learn. This variable is helpful in sum-
marizing the overall performance of our model in terms of
correctly predicting the target output for our input data.
```

We just used `scikit-learn's LinearSVC` model to classify tweets as either "anti-vax" (label 2) or "not anti-vax" (label 0). By training our model with a predefined set of tweets that were manually labeled, we enabled it to learn the context, tonality, and usage pattern of certain keywords within tweets.

Evaluating a Supervised Machine-Learning Algorithm

After training and evaluating how well the model did on predicting the answers it already saw in the training set, the next step is to use our model to predict the labels of our test data, data it has not encountered before.

These predictions are then compared against the actual labels. These performance metrics will likely be worse than the metrics we got for the training data and is a better test of how well the model generalizes to unknown examples or, to borrow a phrase in content analysis, the degree to which the machine-learning algorithm is externally valid.

Accuracy

Let's start with the basics, the ratio of correct predictions to the total number of predictions, multiplied by 100. This gives us the accuracy percentage. In Python, "`metrics.accuracy_score()`" is a function that calculates the accuracy of a machine-learning model. The "`accuracy_score()`" function takes two arrays as parameters:

1. "y_true"—This parameter contains the correct labels.
2. "y_pred"—This parameter contains the predicted labels by our model.

For example, in the code snippet below:

```
accuracy = metrics.accuracy_score(y_test, predicted)
accuracy_percent = round(accuracy * 100, 2)
```

The `'y_test'` would be our `'y_true'` and `'predicted'` would be `'y_pred'`.

The "`accuracy_score()`" function returns a float value between 0.0 and 1.0 indicating a proportion of correctly predicted labels to the total labels. To convert this into a percentage, we multiply the "`accuracy`" value by 100. Finally, to make our output more readable, we round off our "`accuracy_percent`" to two decimal places using Python's built-in function "round()."

Avoid the accuracy trap. In industry, especially in ad tech, I hear folks throw around phrases like "94% accurate" and, while that may sound amazing, that's not enough evidence. Remember, accuracy is only appropriate to report when all classes are balanced. If we knew that approximately half of the tweets were "anti-vax," then accuracy makes a good bit of sense. Otherwise, other metrics, such as precision, recall, and F1 score, provide better insights into a model's performance. If we were looking for a rare behavior, like political talk in a corpus of Facebook posts, approximately 99% of the content would be not political. This means that if my machine-learning algorithm predicted that every post was not political, it would be 99% accurate, but completely worthless, as it would surface no political talk.

Recall

Recall, also known as sensitivity or true positive rate, is a performance metric used in machine learning to evaluate the capability of a classification model

to correctly identify all relevant instances (Powers, 2011). In other words, out of the total true positives, how many instances were correctly labeled by the model?

Mathematically, recall can be defined as the ratio of true positives (TPs) to the sum of true positives and false negatives (FNs; Sokolova & Lapalme, 2009).

```
Recall = TP / (TP + FN)
```

In this formula, a true positive (TP) means that the model correctly predicted a positive case as positive. On the other hand, a false negative (FN) signifies that the model incorrectly predicted a positive case as negative (Kuhn & Johnson, 2013).

For instance, in our text classification example of identifying "anti-vax" sentiment in tweets, a true positive would mean that the model correctly identifies a tweet as "anti-vax." A false negative would mean that a tweet that is actually "anti-vax" is incorrectly classified by the model as "not anti-vax." High recall, therefore, reveals that the model is adept at finding all the "anti-vax" tweets (Chawla et al., 2002).

The recall metric is particularly useful in situations where the cost of false negatives is high. Consider the Apple Watch and its electrocardiogram. It never ceases to amaze me what these watches can do. Apple uses a machine-learning model to identify heart anomalies in a user's daily life. Unlike typical machine-learning problem scenarios, in this particular use case the potential consequences of false negatives are grave—a person with a heart condition is not alerted if the condition is falsely predicted as a non-anomalous heart rate pattern, potentially leading to devastating health consequences or, in severe cases, even death (Powers, 2011). On the other hand, a false positive (predicting a heart anomaly where there is none) could lead to unnecessary anxiety and medical intervention, but it would ultimately be less disastrous.

In this instance, recall becomes critically important. Recall, in the context of this scenario, determines the proportion of actual heart anomalies that the model correctly identifies out of all the actual heart anomalies (Kuhn & Johnson, 2013). A high recall means that most heart anomalies were identified, and users were adequately alerted, thereby potentially saving lives through early detection and treatment (Chawla et al., 2002).

Imagine there are 1,000 Apple Watch users who actually have heart anomalies, and the model only predicted 800 correctly as having heart anomalies, while it reported the remaining 200 users as healthy. The recall, in this case, would be 800 / (800 + 200) = 0.8 or 80%. The remaining 20% that the model failed to recall represents the people who had heart anomalies but were not alerted by their Apple Watch.

Precision

Precision is a performance metric used in machine learning that evaluates the ability of a classification model to correctly label an instance as positive from all the instances it has labeled as positive (Fawcett, 2006). In other words, it is the ratio of correctly predicted positive instances to the total predicted positive instances.

Mathematically, precision can be defined as the ratio of true positives (TPs) to the sum of true positives and false positives (FPs; Powers, 2011).

```
Precision = TP / (TP + FP)
```

In this formula, a true positive (TP) refers to a positive class instance that the model correctly predicts as positive. False positive (FP), on the other hand, refers to a negative class instance that the model wrongly predicts as positive (Sokolova & Lapalme, 2009).

For example, in our text classification task for identifying "anti-vax" sentiment in tweets, a true positive would mean the model correctly identifies an "anti-vax" tweet. A false positive, however, would mean a tweet that's actually not "anti-vax" is incorrectly classified by the model as "anti-vax." A high precision, therefore, implies that most of the tweets the model labels as "anti-vax" are indeed "anti-vax."

Precision becomes particularly important in scenarios where the cost of false positives is high. For instance, in a spam-detection system, a false positive would mean a legitimate (non-spam) email being incorrectly classified as spam, potentially causing the user to miss crucial information. High precision, in this case, ensures that most of the emails flagged as spam are indeed spam, minimizing the chances of useful emails ending up in the spam folder (Fawcett, 2006).

Nevertheless, precision alone does not paint the complete picture of a model's performance. A model might achieve high precision by only predicting a positive class when it is completely sure, leading to many positive instances going undetected (e.g., low recall). As such, precision is often used in conjunction with other metrics like recall and accuracy to get a more holistic view of the model's performance (Sokolova & Lapalme, 2009).

F1

To address this trade-off, the F1 score provides a single value that represents the harmonic mean of precision and recall. It effectively captures both false positives (those incorrectly labeled as positive) and false negatives (actual positives that the model missed), giving a consolidated view of the model's performance (Powers, 2011).

Mathematically, the F1 score is defined as

```
2 * (precision * recall) / (precision + recall)
```

(Sokolova & Lapalme, 2009).

The F-score is referred to as a blended metric because it combines precision and recall into a single number. It gives equal weight to both measures and maximizes when both precision and recall are high. It is useful when you need a balance between precision and recall and there is an uneven class distribution in the dataset. In situations where both false positives and false negatives are costly, an F1 score serves as a more suitable evaluation metric (Powers, 2011).

The F1 score ranges from 0 to 1 where a score of 1 indicates perfect precision and recall, and 0 indicates that either the precision or the recall (or both) is zero. Hence, the higher the F1 score, the better the performance of the model. To calculate the precision, recall, and F1 score metrics in Python, we can use the respective functions provided by the sklearn.metrics module (Pedregosa et al., 2011).

Here's a simple example:

```
from sklearn import metrics

# Let's assume that y_test are the true labels and predict
ed have the labels predicted by the model
y_test = [0, 1, 1, 0, 1, 1]
predicted = [0, 1, 0, 0, 1, 1]

# Calculate Precision
precision = metrics.precision_score(y_test, predicted)
```

While precision offers critical insights into a model's performance, it should ideally be used in conjunction with other metrics like recall or F1 score.

```
# Calculate Recall
recall = metrics.recall_score(y_test, predicted)

# Calculate F1 Score
f1 = metrics.f1_score(y_test, predicted)
```

In our study, the use of the F1 score in evaluating the tweet classification model would give us a balanced perspective on the model's performance. A high F1 score would suggest that the model is reliable in predicting an "anti-vax" tweet (high precision) and is generous enough to catch most "anti-vax" sentiments in the dataset (high recall). Conversely, a low F1 score may suggest potential issues with the model's precision and/or recall that would require further investigation.

Interpreting Performance Metrics

Assessing the adequacy of precision and recall, like many tasks related to model evaluation in machine learning, is largely problem-dependent and strongly tied to the specific context and objectives of the analysis. However, several common heuristics can serve as general guideposts:

1. **Understand the problem context**: One of the first key steps is to fully understand the problem at hand and the relative costs of false positives (which impact precision) and false negatives (which influence recall). As discussed in a previous chapter, political talk is a relatively rare thing to observe on social media. Given that, it would make sense to prioritize a system that uncovers as much of it as possible. If the actual number of political posts/documents that your algorithm ends up surfacing is quite low, you can manually review them and kick out the ones you don't like. This is better than missing posts/documents, and quite easy to manage.
2. **Label, train, and repeat**: Label a few hundred posts/documents by hand and then try to train an algorithm. Odds are, it will be bad, but it's important to get a baseline of just how bad. There is no magic number to say when an F1 score is good enough, but one thing you can do is label/code/annotate your data in batches and try building the model after each additional batch. You should see an improvement. If not, you likely have a poorly defined codebook/codes or poorly labeled data. At some point, even after coding a new batch, your model will improve very little. That's a good time to call it a day on the manual labeling process.
3. **Industry/academic standards**: Published studies in similar domains may reveal precision and recall benchmarks, or "industry standards," that are considered "good enough." These standards could serve as helpful targets. In general, I try to only use machine-learning models with F1s above .85, but this is absolutely a heuristic. More complex concepts will invariably mean lower F1 scores.

Model evaluation metrics should always be considered in the context of the problem and should not be seen as absolute measures of a model's worth. The choice between prioritizing precision or recall is a decision that requires understanding the problem domain, the data, and the specific implications of false positives and false negatives. Underlying any computational social science research, especially when using machine-learning models for classification tasks like the "anti-vax" sentiment analysis, is an important consideration—the tolerance for error. This includes both the ability to correctly identify true positives (precision) and to avoid false negatives (recall). This trade-off is defined by not just the model's capabilities but also the implications of misclassifications.

Imagine your model is being used to classify tweets as "anti-vax" or "not anti-vax." Each misclassification would skew your understanding of public sentiment towards vaccinations. A high rate of false positives (classifying not "anti-vax" tweets as "anti-vax") would overestimate the anti-vaccine sentiment and might misinform policy decisions or public health communication strategies. Therefore, understanding and deciding an acceptable error threshold upfront becomes vital.

This threshold is sensitive to the behavior in question's rarity. If anti-vaccine sentiment is indeed very rare, a model with moderate or even high recall but lower precision might be a reasonable choice. The model would cast a wide net to capture most of the rare "anti-vax" instances, even at the risk of including some false positives.

Reporting Best Practices

To ensure effective communication and reproducibility in scientific reporting of machine-learning classification models, it is important to follow certain best practices when writing up your method section.

1. **Data and preprocessing**: Always describe the source of your dataset and justify its relevance to the problem you are trying to address. Include details about how the data was collected, the size of the dataset, and its distribution. Preprocessing steps such as handling missing data, feature engineering, and text preprocessing (if applicable) should be clearly documented (García et al., 2015; Kuhn & Johnson, 2019).
2. **Model description**: Detail the type of model that you used, including the reasons for your choice. Explain the model's architecture and working principles as simply as you can without getting into mathematical formulas or complex discussions.
3. **Training procedure**: Explain how the model was trained. Details such as the division of data into training, validation, and test sets, the use of specific optimization algorithms, batch size, learning rates, the number of epochs, regularization techniques, and early stopping criteria should be included. I'll discuss these things more in the deep-learning chapter (Bengio et al., 2017).
4. **Performance metrics**: Clearly define the metrics used to evaluate your model. Common metrics for classification problems are accuracy, precision, recall, F1 score, and Area Under the Receiver Operating Characteristic Curve (AUROC). Furthermore, the reasons for choosing these metrics should be well articulated. If custom metrics are used, provide a detailed explanation of their computation (Chicco & Jurman, 2020).
5. **Model evaluation**: Discuss your model-evaluation methodology. It could include techniques like cross-validation, bootstrapping, or holdout validation. Also, the choice of a particular methodology should be justified.

6. **Discussion**: Reflect on your model's performance. Discuss the possible reasons behind the observed results and compare them with the existing state-of-the-art models. Offer insights on the strengths and shortcomings of your model, any observed trends or peculiarities, scopes for future work, and the implications of your findings.
7. **Ethics and bias**: Address potential ethical considerations and biases in your model, specifically when dealing with sensitive data. Discuss the steps taken to safeguard privacy and fairness, and critically evaluate the possible societal impacts of your work.

Remember, the purpose of following these best practices is not merely to comply with editorial or peer-review requirements but to enhance the transparency, credibility, and usability of your work. By clearly reporting your machine-learning models, you are responsibly and effectively contributing to the progression of science.

Model Workshopping

Each model has its strengths and weaknesses, as well as assumptions that may or may not align well with the structure of your dataset. As such, the concept of "one size fits all" does not apply. Identifying the most appropriate model for your problem is not a straightforward task, and it should be treated as an integral part of the machine-learning pipeline.

Model workshopping involves the comparison and examination of different models on your dataset. It's a process that includes planning, executing, analyzing, and iterating various model families, hyperparameters, feature selections, and even preprocessing methods. This iterative approach makes it possible to find the model that provides the best fit for your data according to the performance metrics you consider most relevant.

The goal is not merely to improve prediction accuracy but to enhance the generalizability of your model and better understand the underlying patterns within the data. Therefore, investing time in model experimentation is not only about improving the performance metrics but also gaining valuable insights into your data and the learned model itself. This is particularly significant in computational social science research, where the aim is not merely to predict outcomes but to deliver insights into social behaviors and phenomena.

I strongly encourage you to workshop several machine-learning models to adopt the one that fits your data the best. In the past, I have used machine-learning workbenches with easy-to-use interfaces like DataRobot[4] and Alteryx.[5] These tools make it easy to try different supervised machine-learning algorithms with little effort. I still recommend using these tools if you're new to Python because it gets you quickly to statistical benchmarks. The models that you build and use should perform approximately as well as these workbenches.

However, if I may assume that you are a brave Python coder, in the next portion of the chapter, let's use sci-kit learn to workshop models that are easily applied in Python. We can easily workshop and discover what models work best for us, that is the one that labels the data most like the training data.

Luckily, when it comes to Python code and machine learning, there are a ton of readily available notebooks you can use to get started.[6] If you're a bit more advanced in Python, take some time to review the lecture notebook entitled '`sci-kit_text_classification_workshopping`' for this chapter and try to adapt it to one of your problems.

Closing Thoughts on Supervised Machine Learning

As you will soon find in the chapters that pertain to deep learning, another major factor when selecting a supervised machine-learning model is the training and prediction times. Luckily, most CPUs are quite good enough to run these basic models quickly, and a notebook environment like Google Colab's basic CPU instance should be fine.

In scenarios where quick results are necessary look no further than using sci-kit learn models like the ones we've workshopped here. Later in this book, we'll try more powerful generative artificial intelligence. While those models generally classify text better, they are 10–100x slower and require specialized GPU computing.

The process of model selection should also account for the complexity of the model. Simpler models are often preferred because they're easier to understand, interpret, and communicate. They're also less prone to overfitting. This is known as the principle of Occam's razor (Sober, 2015): among competing models that explain the data equally well, choose the simplest. You may be tempted to use an "ensemble" or "blender" model that combines a myriad of algorithms and slightly outperforms others, but understand that the more complex the model, the more things that can go wrong.

As we transition to the next stage of our computational journey, we will focus on more powerful models that have successfully transcended many of the challenges we've grappled with thus far. These models come with an unprecedented advantage—an existing knowledge of English-language intricacies.

These models have been pre-trained on massive text corpora, enabling them to implicitly understand the nuanced patterns, structures, and rules of English. In essence, they encapsulate a "computer brain" that has already been subjected to rigorous language training, equipping them with a deep and sophisticated understanding of English.

Harnessing these pre-trained models can significantly streamline our machine-learning process. Instead of starting from scratch, we are leveraging a model that's already familiar with the language's complexities. This

paradigm makes the learning process more efficient, often resulting in improved model performance.

In the upcoming chapter, we will delve into the world of these pre-trained models. We will unpack how they are created, why they work so well, and how we can fine-tune them for our specific tasks.

Notes

1 Similar models for text classification can be implemented using the Scikit-learn library. Scikit-learn provides LinearSVC and LogisticRegression which have similar functionality to LibLinear's linear SVM and logistic regression respectively. Scikit-learn has a strong integration with the Python scientific stack and places a heavy emphasis on ease of use, making it a convenient choice for data scientists and researchers.
2 Please note that you need to have the `"forml.xlsx"` file in your current working directory, or else you'll need to specify the appropriate path to the file.
3 A typical proportion in practice, though a rule of thumb is that as you increase your *n*, you may decrease your training set percentage, like we would in sampling a big dataset.
4 For a free trial: https://www.datarobot.com/trial/
5 For a free trial: https://www.alteryx.com/alteryx-analytics-cloud-platform-trial
6 It has been adapted from the work of authors Prettenhofer et al. (n.d).

References

Alpaydin, E. (2020). *Introduction to machine learning* (4th ed.). MIT Press.

Baeza-Yates, R., & Ribeiro-Neto, B. (2011). *Modern information retrieval: The concepts and technology behind search* (2nd ed.). Addison-Wesley.

Bengio, Y., Goodfellow, I. J., & Courville, A. (2017). *Deep learning*. MIT Press.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. https://doi.org/10.1613/jair.953

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, *21*(1), 6. https://doi.org/10.1186/s12864-019-6413-7

Fan, R. E., Chang, K. W., Hsieh, C. J., Wang, X. R., & Lin, C. J. (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, *9*(Aug.), 1871–1874.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, *15*(1), 3133–3181.

García, S., Luengo, J., & Herrera, F. (2015). *Data preprocessing in data mining*. Springer. https://doi.org/10.1007/978-3-319-10247-4

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. http://www.deeplearningbook.org

Han, J., Pei, J., & Kamber, M. (2011). *Data mining: Concepts and techniques* (3rd ed.). Morgan Kaufmann.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An introduction to statistical learning: With applications in R* (2nd ed.). Springer. https://doi.org/10.1007/978-1-0716-1418-1

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the 14th International Joint Conference on Artificial intelligence*, *2,* 1137–1145.

Kotsiantis, S. B. (2007). Supervised machine learning: A review of classification techniques. *Informatica*, *31*(3), 249–268.

Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling.* Springer. https://doi.org/10.1007/978-1-4614-6849-3

Kuhn, M., & Johnson, K. (2019). *Feature engineering and selection: A practical approach for predictive models.* CRC Press.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval.* Cambridge University Press.

McHugh, M. L. (2012). Interrater reliability: The kappa statistic. *Biochemia Medica*, *22*(3), 276–282. https://doi.org/10.11613/BM.2012.031

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(2011), 2825–2830.

Powers, D. M. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, *2*(1), 37–63. https://doi.org/10.9735/2229-3981

Prettenhofer, P., Grisel, O., Blondel, M., Amor, A., & Buitinck, L. (n.d.). *Classification of text documents using sparse features.* Scikit-learn. https://scikit-learn.org/stable/auto_examples/text/plot_document_classification_20newsgroups.html

Provost, F., & Fawcett, T. (2013). *Data science for business: What you need to know about data mining and data-analytic thinking.* O'Reilly Media.

Sober, E. (2015). *Ockham's razors: A user's manual.* Cambridge University Press.

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, *45*(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

Tomeny, T., Vargo, C., & El-Toukhy, S. (2017). Geographic and demographic correlates of autism-related anti-vaccine beliefs on Twitter, 2009–15. *Social Science & Medicine*, *191*, 168–175. https://doi.org/10.1016/j.socscimed.2017.08.041

Chapter 5

Text Classification of News Media Content Categories Using Deep Learning

In the previous chapter, we classified tweets related to vaccines based on whether they propagated "anti-vax" or "not anti-vax" sentiment. This exercise, while challenging, unearthed the potential of supervised machine learning and its capacity to decipher intricate patterns in unstructured data like text.

As we continue our computational journey, let's turn our attention to a new classification task, that of contextual advertising. Imagine working at a media buying company, Chrishare, catering to a client, Theragun. Theragun has an acute understanding of its market—it knows that consumers who value health and wellness are likely to buy their product. Therefore, their marketing strategy is aligned to target this specific consumer segment. The aim? To identify as many news articles that mention health and wellness as possible and display their advertising content alongside them. This is contextual advertising: tailoring advertisements based on who the consumer but the context in which the advertisement is displayed.

You, a computational scientist at Chrishare, have been tasked with the objective of building a *deep- learning algorithm* that can scan news stories and accurately predict the probability that the story is about health and wellness. Drawing upon an open-sourced dataset of approximately 200,000 news headlines between 2012 and 2022 from *HuffPost*, your challenge is not just about harnessing a machine-learning model for prediction but also about contributing towards an efficacious marketing campaign for Theragun (Misra, 2022).[1]

This classification task, however, comes with its unique challenges and requirements. Unlike the previous exercise where the category of interest was quantitatively balanced with the other category, the current task deals with an imbalanced dataset. Health and wellness articles constitute only a fraction of the wide array of categories found in the news and therefore are a minority class in this dataset. This imbalance may introduce bias in our machine-learning model, compelling it to predict largely in favor of the

DOI: 10.4324/9781003514237-5

dominant class. Therefore, addressing this issue becomes central to building an accurate and trustworthy model.

Moreover, the scope of this task extends beyond precision. Hypothetically, it might be tempting to lean towards a model that labels more articles as health and wellness, thus providing a larger pool of articles for Theragun to advertise on. Nonetheless, irrelevant or inaccurate labeling can harm Theragun's credibility and may not yield the desired marketing outcome. Thus, a model that maintains a balance between finding enough relevant articles and ensuring that the articles identified are indeed appropriate is what we want.

In this chapter I will utilize the power of deep learning and modern libraries like TensorFlow and Keras with a wrapper library, ktrain, to develop our text-classification model (Maiya, 2023). The ktrain library simplifies the process of building and tuning deep-learning models. By using it, I aim to find a delicate balance between various evaluation metrics, specifically precision and recall, to ensure that our model not only identifies a sufficient number of health and wellness articles but also identifies predominantly includes genuinely relevant articles.

In the previous chapter, I employed supervised machine-learning techniques, such as SVC and various linear classifiers for document classification (James et al., 2013). While the algorithm successfully classified tweets based on their sentiment, we found that these models had limitations, especially when dealing with linguistic nuances found in natural language (Brownlee, 2017). The two previous methods we've discussed—keyword spotting and traditional bag-of-words models—often fall short of capturing the rich context embedded in language, leading to potential misclassifications.

From Bags of Words to Deep Learning

Given what we've learned, a tempting approach for contextual advertising would be to create an extensive list of health- and wellness-related keywords and define features based on their presence or frequency in the documents. Yet, such an approach might lead to suboptimal results given the complexity of language and the diverse ways health and wellness topics can be discussed. It's unlikely that we could easily arrive at a list that would represent all the different ways health and wellness are talked about in today's news.

Let's tackle these complex language intricacies and capture appropriate contextual meanings, using a better approach—using pre-trained models, specifically word embeddings. Word embeddings are a type of word representation that allows words with similar meanings to have similar representation, thereby capturing semantics, morphological, syntactical, and co-occurrence information of words in a language (Mikolov, Chen, et al., 2013).

Word embeddings are learned in an unsupervised manner from large amounts of text data, which means they can represent a wealth of latent

linguistic information present in the data (Lecun et al., 2015). Therefore, they are highly versatile and can be applied to a variety of tasks in natural language processing, ranging from text classification and clustering to entity recognition, sentiment analysis, and machine translation, performing significantly better than traditional methods.

One of the most prevalent and efficient techniques to generate word embeddings is using pre-trained models such as Word2Vec, FastText, and BERT. These models have already been trained on massive text corpora and can encode comprehensive semantic and contextual information about words. Their advantage in the current context lies in the fact that these pre-trained models "understand" English and its intricacies, transferring this understanding to text classification tasks (Mikolov, Chen et al., 2013).

In simple terms, word embeddings map words in the vocabulary to vectors of real numbers (Pennington et al., 2014). Unlike the traditional bag-of-words (BoW) techniques, the strength of word embeddings lies in their capacity to capture and retain numerous dimensions of information about a word, including its meaning, context, or relationship with other words.

Word embeddings are based on the distributional hypothesis, which postulates that words occurring within similar contexts possess similar meanings. By leveraging this idea, word embeddings can capture multiple dimensions of a word, including its semantic meaning, syntactic role, and contextual usage (Harris, 1954). Instead of treating words as independent units, word embeddings capture contextual and semantic similarities among words (Mikolov, Sutskever, et al., 2013). In our current application, using word embeddings will allow the model to understand, for instance, that "exercise," "workout," and "yoga" are related concepts and belong to the context of health and wellness, even without explicitly programming such associations (Devlin et al., 2018). This also enables the model to consider nearby contexts. For instance, given the word "warrior," it can understand the meaning based on surrounding words, and recognize that it refers to a yoga pose rather than the more common definition.

They also exhibit interesting algebraic properties. They can be manipulated mathematically to reveal meaningful semantic relations between words. For instance, if we take the embedding for "king," subtract the embedding for "man," and add the embedding for "woman," we get a result very close to the embedding for "queen" ("king" – "man" + "woman" = "queen"), signifying the gender relation between the words (Mikolov, Sutskever, et al., 2013).

All things considered, word embeddings offer a continuously valued, dense, and low-dimensional representation of words, which can lead to more nuanced and context-aware natural language processing models (Pennington et al., 2014).

Bag-of-Words Approaches vs. Deep-Learning Approaches

The bag-of-words (BoW) model involves representing the text data as a "bag" of its words, disregarding grammar, part of speech, and word order but keeping the multiplicity of words. The initial steps include text preprocessing methods like tokenization, stop word removal, and lemmatization. After that, the words in the document are represented as feature vectors (see Chapter 4 for a review).

To label a document as "politics" or "not politics," we first need to label a set of training documents, and then feed them to a supervised machine-learning algorithm like the many contained in scikit-learn. From that, the algorithm would surmise a list of politics-related keywords such as "government," "election," "#election2020," "democracy," "republic," and "president," based on their co-occurrences with positive examples from our training dataset. To classify new documents, we simply add up the likelihood of each word in the document correlating with our class of interest, and if the article mentions these keywords above a certain threshold, the document is labeled as "politics"; otherwise, it is categorized as "not politics" (Harris, 1954).

While this approach works decently for our task, it has limitations. Firstly, the BoW model does not consider the semantics and context of the words. Moreover, rare words or unique expressions used in political discourse can be overlooked (Harris, 1954).

Word embeddings offer a solution to these challenges. Here, words are represented as vectors in high-dimensional space where the semantic relationships between words are preserved. Words with similar meanings are located close to each other in this vector space (Mikolov, Chen, et al., 2013).

Unlike the BoW model, the word embedding approach is dynamic and flexible. It can adapt to the evolving linguistic styles in news articles, understand synonyms, and detect politics-related content even when it is not explicitly mentioned using standard political keywords.

Training a Deep-Learning Algorithm

Training a deep-learning model involves the same basic idea of supervision that was discussed in the last chapter. The model, like a student, starts with little knowledge about the subject at hand. The training process involves "teaching" the model by providing it with a large amount of labeled data (Goodfellow et al., 2016).

This process is repeated iteratively on all the examples in the training set. Over time, the model "learns" from the data and gradually improves its predictive accuracy by minimizing the discrepancy between its predictions and the actual labels.

Fine-Tuning Models

Pre-trained models that are smaller and older, like BERT, are typically not used as-is but are fine-tuned or adapted to a specific task. Fine-tuning involves continuing the training process on the specific task data so that the model can adapt its previously learned knowledge to the new task (Chollet, 2015). Fine-tuning is particularly advantageous when the new task has limited labeled data, and as such this is generally how I approach using them for content analysis.

In a nutshell, pre-trained models embody a form of transfer learning, wherein the knowledge gained from one machine-learning task is used to solve another related task. These models have revolutionized many fields in artificial intelligence, particularly in natural-language processing and computer vision, by providing robust, ready-to-use models capable of achieving high performance with lesser amounts of task-specific data.

TensorFlow and Keras

In this chapter, we'll leverage packages built on top of the deep-learning framework TensorFlow. One major forte of TensorFlow is its ability to leverage hardware acceleration technologies like GPUs for computation, making it highly efficient and scalable for diverse computational tasks. Furthermore, thanks to its Python-friendly API and vast community support, TensorFlow is a go-to for training these types of networks (Chollet, 2015).

Within the TensorFlow ecosystem, Keras emerged as a high-level Python library designed to simplify the process of building, training, and deploying neural networks. Developed by Francois Chollet in 2015, Keras acts as an interface for the TensorFlow library and offers a more user-friendly and intuitive approach to deep-learning models (Chollet, 2015).

Bidirectional Encoder Representations from Transformers (BERT)

This chapter will focus on using the pre-trained model Bidirectional Encoder Representations from Transformers (BERT). It's been heralded as a revolutionary step in the field of natural-language processing (Devlin et al., 2018).

Following its introduction, BERT has spawned a series of spinoff models designed for specific languages, applications, or constraints. These include ALBERT, an optimized version of BERT that is lighter, faster, and performs better, and RoBERTa, a robustly optimized BERT model that uses a different training approach and has more training data (Devlin et al., 2018).

Transformers

BERT is built upon the Transformer model. Transformers were introduced in the paper "Attention Is All You Need" by Vaswani et al. (2017), and quickly revolutionized the field of NLP by proposing an entirely new approach for processing language.

Suppose we have a news article about a political campaign. The article starts by discussing the candidate's early life and experiences, then moves on to their political career, and finally ends with their current campaign promises and strategies. In this article, there might be a sentence in the last paragraph like, "The candidate's commitment to affordable healthcare is deeply rooted in their early experiences." Here, the phrase "early experiences" refers to information provided way back in the first few paragraphs of the article. For older machine-learning models, it would be challenging to make this connection if the article is long and the reference is several sentences or paragraphs away. This is because these models process language word by word, and their ability to retain information diminishes as the distance increases.

BERT and all Transformer models can handle this type of long-term dependency much better (Vaswani et al., 2017). They process the entire text at once and pay attention to all parts of the input simultaneously, allowing them to maintain the context and make connections between distant parts of the text (Devlin et al., 2018). This makes them particularly effective for tasks like text classification, sentiment analysis, and, from my experience, computational content analyses (Sun et al., 2019).

The fundamental building block of a Transformer is the "attention mechanism," which decides which parts of the text the model should focus on at any given time (Vaswani et al., 2017). At each step of training, the model employs multiple self-attention heads that scrutinize the entire text and determine the importance of every other word in understanding the current word's context. This multiplicity allows Transformers to learn contextual relations between words and deal with subtleties of language such as pronoun-antecedent relationships, irrespective of the distance between the words (Clark et al., 2019).

Implementing Transformers in our text classification workflow translates to simpler preprocessing pipelines. The whole concept of feature selection used to be a daunting choice for me as a researcher. What features should I include in my bag of words model? Unigrams (e.g., housing), bigrams (e.g., housing policy), or trigrams (e.g., proposed housing policy)? Should we attach the part of speech that goes with each word to help the model disambiguate between different uses of words? TensorFlow eliminates this type of feature selection workshopping by providing an authentic understanding of language owing to lesser intervention in its natural form.

Building a Contextual Advertising Classifier

Now that we've defined the various terminologies we're going to use, let's use deep learning to predict the probability of a news article being about health and wellness for the media-buying company Chrishare. The data we will be using for this project comes from the *HuffPost*, sourced from Kaggle, and contains around 210,000 news headlines from 2012 to 2022.[2]

The data consists of news headlines from various categories such as "POLITICS," "WELLNESS," "ENTERTAINMENT," "TRAVEL," "STYLE & BEAUTY," "PARENTING," "HEALTHY LIVING," and more. To address the goal of our problem, we primarily need to identify those news articles that belong to the "HEALTHY LIVING" category. This is our "target class."

I'm choosing to show you ktrain here (Maiya, 2023). It's essentially Keras for dummies but it is not without its downsides. Over time, as Keras updates, if ktrain is not maintained, it could break. Invariably, we lose a lot of control by using a package that is so abstract. However, I've workshopped ktrain against BoW approaches for years and it always fits the best text classification model. Moreover, I haven't managed to break it yet, which is a very good bar to pass. Remember, ktrain is a lightweight Python wrapper for TensorFlow's Keras API, to make the use of BERT less daunting. Ktrain streamlines the process from loading and preprocessing data to building and training models, and finally making predictions.

Let's start with importing the necessary libraries and setting the directories to store and access different datasets and models. Next, we load the dataset into a pandas DataFrame; a DataFrame is a two-dimensional labeled data structure that is most used in data-manipulation tasks in Python.

```
import os
import ktrain
import pandas as pd
ROOT_DIR = "/1_text_classification"
DATA_DIR = "%s" % ROOT_DIR
EVAL_DIR = "%s/evaluation" % ROOT_DIR
MODEL_DIR = "%s/models" % ROOT_DIR #find the data here:
https://www.kaggle.com/datasets/rmisra/news-category-dataset
reviews = pd.read_json('%s/news_category_trainingdata.json'
% DATA_DIR)
```

Once the data is loaded, we need to combine the "headline" and "short_description" columns into a single "combined_text" column because the machine-learning model will be learning from this text.

```
reviews['combined_text'] = reviews['headline'] + ' ' +
reviews['short_description']
```

Next, we create a new column "healthy" and classify an article as "non-healthy" or "healthy", with "1" representing a healthy article and "0" capturing non-healthy ones.

```
reviews['healthy'] = np.where((reviews['category'] ==
'HEALTHY LIVING'), 1, 0)
```

The distribution of articles across these categories is not evenly balanced. The "HEALTHY LIVING" category, which is of sole interest to us, only forms a small part of the entire dataset. This means we have an imbalanced classification problem on our hands, where one class (in our case, "HEALTHY LIVING") is underrepresented. This can potentially bias our model towards the majority classes, resulting in suboptimal identification of our target class (He & Garcia, 2009).

To address this, we can adopt certain techniques such as undersampling the majority classes, oversampling the minority class, or synthetic minority over-sampling (SMOTE) to balance our classes before feeding them to the model. Here, I'm simply ensuring that the dataset is balanced by ensuring that 50% of the training instances are positive examples and 50% are negatives.

```
healthy = reviews[reviews['healthy'] == 1]
sample_amount = len(healthy)
#we're balancing the classes here by ensuring that 0/1 are
the same sample size
not_healthy = reviews[reviews['healthy'] ==
0].sample(n=sample_amount)
review_sample = pd.concat([healthy,not_healthy])
```

Here, we utilize the "`text.Transformer`" utility from ktrain to use the BERT model. To ensure our model doesn't get overwhelmed with extremely long texts, we set a maximum length (maxlen) of 512 tokens, a technical limitation of BERT. Tokens are essentially pieces of the whole text. Generally, they are about four characters, not quite a whole word. So, BERT can handle ~385 words. It's not great when labeling longer texts, but it should work for news articles, which are generally around this size.

```
t = text.Transformer('distilbert-base-uncased', maxlen=512,
class_names=target_names)
```

Converting text data into a format that the model can understand is termed preprocessing. Here, the data is split into "train" and "val" sets for model training and validation purposes, respectively. The majority of the data is used for training and 10% is used for validation.

```
train, val, preprocess = text.texts_from_df(review_sample,
'combined_text', label_columns=['healthy'], val_df=None,
max_features=20000, maxlen=512, val_pct=0.1,
preprocess_mode='distilbert', verbose=1)
```

Deep-Learning Evaluation Metrics

In deep learning, we hear the term "loss" mentioned as a measure of how well an algorithm either 1) fits a training set or 2) fits a test set. While loss can be operationalized to be an array of different statistics, in general, it can be thought of as we would think of error in fitting a regression model. A high loss would mean that the model's predictions of whether a news article is about health and wellness are far from the actual categories of the articles. This would imply that the model is not performing well in identifying relevant articles for Theragun's advertising campaign. On the other hand, a low loss indicates that the model's predictions align closely with the actual categories, suggesting that the model is accurately classifying the articles.

While the loss gives us an idea of how well the model is learning from the training data, the validation loss tells us how well the model generalizes to new, unseen data, that 10% or so we held out earlier (Goodfellow et al., 2016). The validation loss is calculated using a separate set of data (the validation set) that the model has not been trained on.

A model that performs well on the training data but poorly on the validation data is said to be "overfitting." Overfitting is a common problem in machine learning where the model learns the training data too well, including its noise and outliers, and performs poorly on new data (Hawkins, 2004).

In the context of machine learning, and more specifically in training neural network models, an epoch is a term used to describe one complete pass through the entire training dataset (Brownlee, 2017). When you are training a model on your dataset, the model sees and learns from the data in batches. Once the model has seen all the batches once, it has completed one epoch.

During an epoch, the neural network's weights are updated multiple times through an optimization algorithm, such as gradient descent, based on the loss function defined for the problem. With each batch, the model makes predictions, compares them with the actual targets using the loss function, and then adjusts the weights in a way that would reduce the loss. When the model goes through all batches, and hence all the data, it completes the learning cycle, i.e., an epoch.

The number of epochs is a hyperparameter that you, the researcher, manually set. It defines the number of times that the learning algorithm will work through the entire training dataset. Too few epochs can result in an underfit model, whereas too many can lead to overfitting. The ideal number of

epochs is such that the model learns the general patterns in the data without capturing the noise. Monitoring validation loss in addition to training loss is important to avoid overfitting. The moment when the validation loss stops decreasing and starts increasing is often the point at which the model begins to overfit, and typically where we stop training our model. This is known as early stopping, because regardless of how many epochs we define, when early stopping is implemented, the model with the lowest validation loss is considered best and used as the final fit for the model (Prechelt, 1998).

Here we have set "early_stopping" to true, which means the model training will stop if the model's performance on the validation set stops improving for two consecutive epochs. Most basic models will reach a point where the decrease in loss starts to slow down significantly, forming an "elbow" in the loss curve. Beyond this point, the model's performance on the training data may continue to improve slowly, but its performance on the validation data may start to deteriorate. This is a sign that the model is starting to overfit the training data and is losing its ability to generalize to new data. By using early stopping, we can prevent the model from reaching this point and ensure that it maintains its ability to make accurate predictions on new data.

```
history=learner.autofit(1e-4,checkpoint_folder='checkpoint',
epochs=12, early_stopping=True)
```

TensorFlow fitting output includes the "loss" and "accuracy" for each epoch in the training phase as well as the loss and accuracy for the validation data. "Loss," or "log loss," stands for a common loss function for binary classification, also known as "Binary Cross-Entropy loss" (Chollet, 2017). The loss function calculates the disparity between the algorithm's prediction and the actual label. The goal of the model is to minimize this loss value, which implies the model is improving in making accurate predictions (Goodfellow et al., 2016).

Imagine we have two predictions made by our model for a binary classification task. The first sample is class 1 (positive), and the model predicts it with a high probability of 0.9. The second sample is class 0 (negative), and the model predicts it with a probability of 0.2, meaning it leans towards the negative class, which is correct. We calculate the loss for each sample using the binary cross-entropy formula, which involves taking the negative log of the predicted probability if the actual class is 1, and the negative log of the complement of the predicted probability if the actual class is 0 (Nielsen, 2015).

So for the first sample, the loss is the negative log of 0.9, and for the second sample, the loss is the negative log of (1–0.2), which is the negative log of 0.8. If we use rough estimates for these logs ($\log(0.9) \approx -0.11$ and $\log(0.8) \approx -0.22$), the losses would be approximately 0.11 and 0.22, respectively. To find the average loss across both samples, we just take the mean of these two numbers.

Therefore, in our simple example, the average loss for the two predictions would be around 0.17, where a lower number indicates better accuracy of the predictions. This gives us a quick sense of how well our model is doing: the lower the loss, the more accurate our model is at classifying samples (Zhang & Wallace, 2017).

"Accuracy," on the other hand, represents the percentage of correct predictions made by the model. An accuracy of 0.8 or 80% suggests that the model is predicting the right class 80% of the time. In the case of training accuracy, it signifies the model's performance on the training dataset, while validation accuracy is the performance metric for the validation dataset (Sokolova & Lapalme, 2009).

Finally, we validate the model using the "`learner.validate()`" function. This function uses the validation dataset to assess the performance of the trained model. It outputs the precision, recall, and F1-score, giving a comprehensive understanding of the model's ability to identify "healthy" articles accurately and exhaustively (Powers, 2011).

```
validation = learner.validate(val_data=val, print_
report=True)
```

Given this was a binary classification problem, we should report precision, recall, and F1 scores when writing this up for a manuscript. Remember:

1. **Precision**: Precision is a metric that quantifies the ability of a model to identify only the relevant instances. Simply put, it calculates the proportion of correctly predicted positive observations out of the total predicted positives. Precision is a good metric to use when the cost of a false positive is high.
2. **Recall**: Recall, also known as sensitivity or true positive rate, measures the ability of a model to find all the relevant cases within a dataset. It calculates the proportion of actual positives that were correctly predicted by the model. Recall is a good metric to use when the cost of missing a positive instance (false negative) is high.
3. **F1 score**: The F1 score is the harmonic mean of precision and recall. It acts as a balance between the precision and the recall. F1 reaches its best value at 1 (representing perfect precision and recall) and worst at 0.

Model Interpretability

Model interpretability, a critical aspect of any prediction system, becomes complicated with neural networks due to their "black-box" nature. One of the things that may keep us up at night as social scientists is the idea that we have no idea what these deep-learning algorithms are doing under the hood. While I've gone to a bit of length to explain how TensorFlow and its

corresponding packages work, with deep learning, we've lost the ability to directly inspect our model. In the linear BoW models we unpacked in the last chapter, we can clearly see the coefficient weight of terms. This allows us to qualitatively understand how the machine arrives at its classifications. A BoW prediction is as simple as adding up a bunch of weights of words. Words with large coefficients drive prediction for the positive class, and as such, a quick scan to make sure they intuitively align with our concept is a good practice to ensure your model is classifying the ways you would expect it to.

Unfortunately, the high dimensionality of deep-learning data makes it impossible to intuitively represent simple coefficients. To address this concern, SHapley Additive exPlanations (SHAP) are a unified measure of feature importance that allocates the contribution of each feature to the prediction for each instance. In other words, SHAP values interpret the impact of having a certain value for a given feature in comparison to the prediction we'd make if that feature took some baseline value.

The challenge with interpreting a deep-learning model, such as BERT, in a similar manner, is that the impact of a feature (like a specific word in our case) on the model's prediction is context-dependent and nonlinear (Vaswani et al., 2017). The impact of a word cannot be measured in isolation as it can vary depending on the context, that is, other words present in the text.

SHAP values attribute to each feature the change in the expected model prediction when conditioning on that feature (Lundberg & Lee, 2017). For example, consider a SHAP value for the word "exercise" in a news article categorized under "HEALTHY LIVING." The SHAP value for "exercise" assesses the contribution of this word to the final prediction, conditioning on the presence of other words in the article. A high SHAP value would suggest that the word "exercise" plays a crucial role in classifying a news article as "healthy" or "non-healthy." Therefore, similar to a regression coefficient, the SHAP value provides a numerical value signifying the "importance" or "weight" of a feature in the final prediction.

However, unlike a simple linear model where the coefficient remains the same for all instances, the SHAP value for the word "exercise" would accurately capture its varying importance in different articles. In an article about fitness regimes, "exercise" might have a higher SHAP value than in an article about mental health, thereby encapsulating the context-dependent and intricate nature of language.

```
article = "Boulder, CO – The University of Colorado,
Boulder's own Professor Pat Ferrucci was honored this week
with a prestigious award recognizing him as the top fitness
enthusiast on campus. Ferrucci, renowned for his scholarly
accomplishments and teaching excellence, stunned people
with his dedication to fitness. He logged an astonishing
```

```
300 hours at the university gym in just one semester.
Evidence? The BuffCard's numerous swipe-ins, making the
turnstiles dizzy. According to the BuffCard data, Ferrucci
single-handedly accounted for nearly 10% of total gym
check-ins. Stay healthy, train hard, be like Ferrucci!"

# Load the predictor from the trained model and preprocessor
predictor = ktrain.get_predictor(learner.model, preprocess)
# Use the predictor to explain the article's classification
predictor.explain(article)
```

You can interpret the output as the influence of each word on the model's prediction. Just as coefficients exhibit the magnitude and direction of the relationship between an independent variable and the dependent variable in a traditional regression analysis, SHAP values allow us to understand the contribution of each word in driving the model's predictions in our text classification task.

However, if we're honest, deep-learning models do often act as "black boxes," providing very little interpretability or insight into their decision-making processes at scale. SHAP values can't really be generalized for a model like we can coefficients due to their context dependency. This lack of transparency could be problematic in scenarios where interpretability is critical, such as in healthcare, finance, or legal settings where stakeholders may require understandable explanations for predictions. For us social scientists, I put forward that the intercoder reliability checks that we conduct from human to machine address the bulk of this concern. We have "gold standard data," that is data that has ground truth labels. If we can be sure that we've compared a robust enough sample of our data between humans and computer, and we see no odd or unusual misclassifications, we can rest easier knowing the computer is doing what we set it out to do.

Deep Learning, Big Data

BERT is pre-trained and has some pre-baked knowledge of the human language, but that does not mean it works well with small data. Once, a senior FAANG engineer told me that TensorFlow models with BERT required "tens of thousands" of training documents to fit a problem well. In my experience, if the data is well labeled (that is, if you've done a good content analysis), it's more like thousands, not tens of thousands. However, it still takes quite a bit of time to label a few thousand news articles.

Also, be warned that all deep-learning involves a training process that is computationally intensive and time-consuming. Linear BoW models may take seconds to train and use, BERT is currently minutes to hours.

High-end graphic processing unit processors (GPUs) are required to use ktrain promptly. You could buy an NVIDIA RTX 4090 (MSRP $1,599) and run these on a desktop computer in your office. Or, you could be sane and buy a $10 to $50 monthly subscription to access similar GPUs on Google Colab. Be warned, Google Colab now essentially charges per hour, and runtimes are limited to about a day—meaning, you can't train for more than a day. In my experience, smaller, hand-labeled datasets are not going to take this amount of time but understanding that GPU computing is the precious resource that everyone pays for, one way or another, is important. I've tried running basic TensorFlow algorithms on my M2 Max Macbook with little success. Integrated consumer graphic chips, no matter how nice they are, are just not built for these types of problems. This becomes evident when you see that the size of a 4090 card is about the size of a loaf of bread.

Understanding Probabilities

As we talked about in the evaluation section of this chapter, machine-learning algorithms, whether they are BoW or deep learning, rarely ever return labeled data. Instead, they return probabilities that a text matches a specific class or a set of classes. I'll close this chapter by introducing an automated content analysis that we did and showing you the probabilities that were generated for that data as they pertain to incivility, broadly.

In 2018, the U.S. House of Representatives Permanent Select Committee on Intelligence released roughly one gigabyte of Portable Document Formats (PDFs). Each file contained one advertisement (a social media post that was given additional impressions in exchange for money). These advertisements garnered some press coverage because they were uncovered to be orchestrated by the Russian government as an attempt to spark outgroup incivility (Vargo & Hopp, 2020). In other words, the Russians did a bunch of "othering" in the 2016 election on Meta's platforms, trying to sow discord among U.S. citizens. In all 2,603 ads were downloaded.

To perform data science techniques on the data, the data needed to be converted to a structured format. Unfortunately, and perhaps intentionally, the PDFs were not encoded in a way that would allow the text from the documents to be easily extracted. Textract extracts "information embedded in . . . PDFs, etc—so-called 'dark data'— . . . without any irrelevant markup" (Malmgren, 2014, p. 1). I used Textract to convert the PDFs of an advertisement to text. String and regular expression matching were then used to extract specific fields from the ads, including the body copy (a.k.a., text) of an ad and its metadata (e.g., clicks, cost, impressions, and so on).

In addition, we ran the text of the social media ad through the toxicity classifier that we mentioned in Chapter 3 from Jigsaw's Perspective API, which is built on TensorFlow and Transformers very much like the model we built earlier in this chapter. We have the actual probabilities for each one of

Perspective's models, including "TOXICITY," "IDENTITY_ATTACK," and "INSULTS" (see Chapter 3 for definitions and a discussion of these models).

Validating a TensorFlow-like Classification Model for Use

I'm always of the disposition that even black-box tools like Perspective are not OK by default, as it is a black box. If I see a reviewer use a tool like this without some effort to *externally validate* the tool, I get worried that they are just throwing hammers around a room to see if things stick. In these cases, I ask reviewers to introduce at least one, and preferably more, human coders, and have them label a small sample. For the study at hand, my trusted colleague, Toby Hopp, and I independently labeled 255 randomly selected ads (~10% of the analytic sample). Using the codebook provided in Appendix A, we scored any advertisement containing one or more of the attributes of interest as a 1. If the advertisement did not contain any of the incivility attributes, it was scored as a zero. Here, the coders agreed in 239 of the 255 coded cases (pairwise agreement = 93.7%; Cohen's kappa = .87). This step didn't assess the algorithm, but instead gave us some evidence that our incivility concept and its subconcepts had good validity. That is, two humans could independently review the codebook and make similar decisions.

Having shown that the coding scheme was reliable for human annotators, one of the original coders (me) next re-coded each advertisement in the sub-sample specifically for the presence of identity-based negative language, inflammatory language, obscene language, and threatening language (1= present, 0 = not present). A given advertisement could feature more than one negativity element. These annotations were subsequently compared to the annotation scores generated by the Perspective API. The Perspective API scores text on a *probability* basis, ranging from 0 to 1.

The interpretation of this probability is grounded in the concept of conditional probability in statistics. In the context of text classification, for instance, when an algorithm outputs a probability of 0.8 for a news article belonging to "health and wellness," it is indicating that given the features of this article (which could be its words, phrases, or other textual characteristics), there is an 80% chance that it belongs to the "health and wellness" category.

Welch independent samples t-tests were used to assess the average scores of those advertisements coded as containing the attribute relative to those that were determined not to contain the attribute. The idea is simple, if the algorithms were generally picking up on the concepts, their mean probability scores would be much higher for the set of documents labeled by humans to contain that attribute (e.g., identity-based threats), than documents that do not contain that attribute (e.g., ads with no identity-based threats). Given the significant differences across groups and the large effect sizes associated with each conducted test, we determined that the Perspective API performed in an externally valid manner.

The results of the t-tests showed that this was indeed the case for all four attributes.[3] The advertisements that were manually coded as containing each attribute had significantly higher average scores from the Perspective API than those coded as not containing the attribute. The effect sizes (d) were also calculated for each test. Effect sizes provide a measure of the magnitude of the difference between two groups. In this case, the effect sizes were all fairly large, indicating that the differences in scores between the advertisements containing each attribute and those not containing the attribute were not just statistically significant, but also practically significant.

Turning Probabilities into Classifications

Our study was ultimately interested in whether uncivil content would get more engagement. We opted to directly use the raw probabilities generated by the model for further statistical analysis. These probabilities lie between 0 and 1, and therefore eliminate the need to use logistic regression, while also avoiding the step of losing the confidence associated with the probability scores. In general, the Perspective API is quite good at spotting blatant examples of incivility and associates those classifications with higher probability scores. Transforming a probability score into a binary classification (e.g., hate speech or no hate speech) loses that granularity and treats all content the same.

Setting Thresholds for Binary Classification

However, there will invariably come times when you will want to transform the probabilities into binary classifications by setting a decision threshold. After all, turning back to our contextual advertising example at the start of this chapter, we either can choose to advertise on a website, or not. There is no benefit to a probability, and in fact we must use the algorithm to decide, so a cutoff is necessary. Choosing this threshold is very important and largely up to your discretion.

A common threshold is 0.5. If the model estimates the probability of an article being in the "health and wellness" category as greater than 0.5, you might classify it as a "health and wellness" article; if the estimated probability is less than 0.5, you might classify it as a "non-health and wellness" article.

Such binary classifications can facilitate further analysis. For example, you could compute the confusion matrix, which provides a comprehensive summary of correctly and incorrectly classified articles. Metrics such as accuracy, precision, recall, and F1 score can be derived from the confusion matrix and used to evaluate the performance of the classification. These calculations can't be done with probability scores themselves.

Remember that any transformation of the probabilities into binary classifications involves a loss of information. For example, two articles, one with a probability of 0.51 and the other with a probability of 0.99, would both

be classified as "health and wellness" articles, even though the model is far more confident about the second than the first.

Determining an appropriate cutoff is not always straightforward and largely depends on the cost trade-off between false positives and false negatives. For instance, suppose we are using machine learning to label a rare occurrence, such as whether a Facebook ad contains a racial slur. A false negative (missing a genuine case) may have significantly more severe consequences for Facebook than a false positive (a post being temporarily deleted, which could be appealed). In this case, we might opt for a lower threshold, like 0.4, favoring sensitivity over specificity.

A good practice for rare behaviors is to set your threshold low, and simply review the edge cases yourself. If you have your predicted data in a spreadsheet, find the column associated with the concept of interest, and sort that column by probability score from high to low. You should be able to quickly scan the documents and progress down the range of probability scores until you start to see blatant misclassifications. Manually finding and inspecting that probability score will best separate the wheat from the chaff, as you will be able to visually see precision decay as your probability range drops. Seeing where that approximate range is with your own two eyes is often best.

If we increase the probability cutoff from 0.5 to, say, 0.7, what we are essentially doing is making our model more conservative about assigning instances to the target class. In effect, we are increasing the stringency for class assignment, leading to fewer instances being classified as positives.

This will likely increase precision because we are only classifying instances as positive when the model is more certain (above 0.7 probability), thereby reducing the chance of false positives. However, this comes at the cost of reducing recall because by being more stringent, we are likely to miss instances of the target class that the model is less certain about. So, we could end up with more false negatives, which lowers recall.

Beyond Transformers to Large Language Models

As discussed earlier in this chapter, deep-learning models like BERT have some general sense of what words come next. This general understanding of context helps us label documents better. In the next chapter, we will explore the latest iteration of these models, large language models (LLMs). These models take this concept of masking, or knowing what words best come next, and don't just classify or distinguish between different categories, they can generate entirely new data instances like the data they've been trained on. It's kind of like smashing the most likely button on the auto-suggest texting feature on Android and iOS but with much higher precision and understanding of context. These models learn the underlying probabilistic distribution of the data, enabling them to generate new data that is statistically like their training data.

For tasks like ours, involving text classification and understanding, LLMs open up a whole new array of possibilities. As we'll see soon, these models can potentially help us in "unsupervised" classification tasks. This is the ability to classify or categorize text data when we do not have predefined labels for training, a common challenge in real-world settings where obtaining labeled data can be expensive or impractical.

Not only that, LLMs have shown incredible results in understanding context, capturing sentiment, performing language translation, and even producing creative content like poetry or art, rendering them an exciting field of study in the realm of computational text analysis.

In the next chapter, we will unpack the theory behind these models, how they are constructed, why they work so well, and how we can fine-tune them to suit our specific text classification tasks. We will explore some of the most popular generative models available today and learn how they have begun to redefine what machines can create.

Notes

1 You can review the Kaggle page for the data here: https://www.kaggle.com/datasets/rmisra/news-category-dataset. The data is available to share and adapt under license CC BY 4.0.

2 Rishabh Misra, a machine-learning engineer from San Francisco and a Kaggle Expert, is the author of this data, and I thank him greatly for his generous data sharing license (Misra, 2022).

3 In the case of identity-based negative language, the average attribute score for advertisements coded as 1 (M = .62, SD = .17) was significantly larger than the average attribute scores for those advertisements coded as 0 (M = .29, SD = .23), t = 7.38, df = 174.42, p < .001, d = 1.51. Similarly, advertisements determined by the human coder as featuring inflammatory language had significantly higher mean scores (M = .60, SD = .20) than those determined to not possess the attribute (M = .36, SD = .22), t = 8.81, df = 190.66, p < .001, d = 1.13. The same pattern was true for the threat variable: posts coded as 1 by the human coder (M = .49, SD = .19) had higher computer-assigned scores than those coded as 0 (M = .23, SD = .15), t = 9.61, df = 75.57, p < .001, d = 1.68. Finally, for the obscenity variable, posts scored as 1 by the human coder had higher mean scores (M = .69, SD = .33) than posts scored as 0 (M = .14, SD = .19), t = 4.91, df = 8.20, p < .01, d = 2.74.

References

Brownlee, J. (2017). *Deep learning for natural language processing: Develop deep learning models for your natural language problems*. Machine Learning Mastery.

Chollet, F. (2015). Keras. https://keras.io

Chollet, F. (2017). *Deep learning with Python*. Manning.

Clark, K., Khandelwal, U., Levy, O., & Manning, C. D. (2019). *What does BERT look at? An analysis of BERT's attention*. arXiv. https://doi.org/10.48550/arXiv.1906.04341

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of deep bidirectional transformers for language understanding*. arXiv. https://doi.org/10.48550/arXiv.1810.04805

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Harris, Z. S. (1954). Distributional structure. *Word*, *10*(2–3), 146–162. https://doi.org/10.1080/00437956.1954.11659520

Hawkins, D. M. (2004). The problem of overfitting. *Journal of Chemical Information and Computer Sciences*, *44*(1), 1–12. https://doi.org/10.1021/ci0342472

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263–1284. http://doi.org/10.1109/TKDE.2008.239

House Permanent Select Committee on Intelligence. (2018). *Exposing Russia's effort to sow discord online: The Internet Research Agency and advertisements*. https://intelligence.house.gov/social-media-content/

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning*. Springer.

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett (Eds.), *Advances in neural information processing systems 30* (pp. 4765–4774). Neural Information Processing Systems Foundation.

Maiya, A. (2023). *ktrain: A low-code library for augmented machine learning*. [Computer software]. GitHub. https://github.com/amaiya/ktrain

Malmgren, D. (2014). *Textract documentation: Release 1.1.0*. Read the Docs. https://textract.readthedocs.io/_/downloads/en/v1.1.0/pdf/

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space*. arXiv. https://doi.org/10.48550/arXiv.1301.3781

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In C. J. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K.Q. Weinberger (Eds.), *Advances neural information processing systems 26* (vol. 2, pp. 3111–3119). Neural Information Processing Systems Foundation.

Misra, R. (2022). *News category dataset*. arXiv. https://doi.org/10.48550/arXiv.2209.11429

Nielsen, M. A. (2015). *Neural networks and deep learning*. Determination Press.

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. In A. Moschitti, B. Pang, W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP*; pp. 1532–1543). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1

Powers, D. M. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, *2*(1), 37–63.

Prechelt, L. (1998). Early stopping—but when? In G. Montavon, G. B. Orr, & K. R. Muller (Eds.), *Neural networks: Tricks of the trade. Lecture notes in computer science: Vol. 7700* (pp. 53–67). Springer. https://doi.org/10.1007/978-3-642-35289-8_5

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, *45*(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019). How to fine-tune BERT for text classification? In M. Sun, X. Huang, H. Ji, Z. Liu, & Y. Liu (Eds.), *Chinese computational linguistics. CCL 2019. Lecture notes in computer science: Vol. 11856* (pp. 194–206). Springer. https://doi.org/10.1007/978-3-030-32381-3_16

Vargo, C., & Hopp, T. (2020). Associations between advertisement negativity and political advertisement engagement: A computational case study of Russian-linked Facebook and Instagram content. *Journalism & Mass Communication Quarterly*, *97*(3), 743–761. https://doi.org/10.1177/1077699020911884

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.) *Advances in neural information processing systems*, *30* (pp. 5999–6009). Neural Information Processing Systems Foundation.

Zhang, Y., & Wallace, B. (2017). *A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification*. arXiv. https://doi.org/10.48550/arXiv.1510.03820

Chapter 6

Leveraging Generative AI for Content Analysis

Krippendorff (2022) argues that texts do not merely map or indicate features of an existing world but construct worlds for competent speakers of a language to see, enact, and live within. As you will soon find, generative AI in and of itself contains a world of knowledge. To extend his metaphor, I believe we are still the competent speakers. That is, for a good content analysis, we still must provide the specific language and instruction that we can use to make our analysis come alive.

Large language models (LLMs) and generative AI were born out of the desire of computer scientists to have computers learn the distribution and natural propensities of words (Bengio et al., 2003). All the generative AI we enjoy today, like ChatGPT, runs some type of model (e.g., GPT-4) that was trained on vast datasets, from Wikipedia to news content, and everything in between that can be found on the web (Brown et al., 2020).

LLMs have learned to predict the subsequent item in a sequence, whether it's the next word in a sentence or the next frame in a video (Vaswani et al., 2017). The premise is simple: if the model can be certain what single token comes next, all it must do is repeat this task over and over until a sufficient response has been generated. This "next word problem" is how all LLMs are trained. Imagine I give you a sentence and withhold a single word. You would then have to guess what that word might be based on the context provided by the rest of the sentence. Given that you're a smart person and have read millions of words in your lifetime, you probably can guess what is likely to come next. This is essentially what LLMs do at a much larger scale and with a far more complex array of sentences and contexts. They continuously predict and adjust based on the probabilities of what the next word should be, based on what they've seen in the past from their training data. It turns out, for many simple problems, the best guess of what comes next is usually right.

The uniqueness of generative pre-trained transformer (GPT) models is their capacity to predict the probability of a word given its predecessors

DOI: 10.4324/9781003514237-6

in the text, known as autoregression (Radford et al., 2019). As we talked about with pre-trained word language models like BERT, LLMs generate new sentences by predicting the subsequent word based on the context of the past words (Devlin et al., 2019).

While we know less about the training of GPT-4, training GPT-3 was a massive task due to its unprecedented scale. It consists of 175 billion tunable weights that the model uses to make accurate predictions (Brown et al., 2020). To put this into perspective, its predecessor, GPT-2, comprised 1.5 billion parameters whereas BERT consists of 340 million (Devlin et al., 2019; Radford et al., 2019). The incredible size of newer GPT models is one of its defining features and a crucial factor in its impressive ability to understand context and produce accurate language output. Considering the tremendous size of the GPT-4 model and the sheer volume of data it processed, it is safe to assume that the training spanned months, even when run on hundreds or thousands of high-performance GPUs that cost Microsoft an absolute fortune to use (Brown et al., 2020).

Last chapter we talked about BERT, Google's pre-trained brain that is good at beating BoW approaches to classification (Devlin et al., 2019). Since then, GPT models have burst onto the scene. OpenAI introduced GPT-3 in June 2020 (Brown et al., 2020) and currently GPT-4 is the latest addition to the GPT series, with GPT-5 due possibly even before this book is published (OpenAI, n.d.).

Running Your Own GPUs

With the advancements in technology, graphic processing units (GPUs) that can run these models locally and on the cloud are attainable. NVIDIA, the leading designer and manufacturer of GPUs, recently introduced its latest professional-grade offering: the NVIDIA A100 Tensor Core Graphics Card (NVIDIA, 2020). This GPU can offer up to 80 GB of high-bandwidth memory (HBM2), a memory type specifically catered to high-performance computing. However, this power comes with a significant price tag. The MSRP hovers around $11,000, presenting a considerable expense that isn't feasible for most individual researchers or smaller institutions (NVIDIA, 2020).

Newer LLMs use somewhere between 6 and 72 gigabytes of storage. These models can be, if the author of the model wishes, shared online, and downloaded freely by anyone, including you. However, the kicker to these free brains is that the model itself must fit into the RAM of your GPU. An RTX 4090 can process data through an LLM quickly, but it only has 24 GB of RAM. As a result, and at the time of writing this, it can only run models

that have 13B parameters, not ones that have 70B (NVIDIA, 2020). Can you guess which model is generally better at doing content analysis? Bigger is usually better. This means to fit a 70B model, you would have to pony up $6,800 to buy a NVIDIA RTX 600 Ada. The big difference? 48GB of RAM, which will just fit a properly quantized and fine-tuned 70B LLaMA 3 model, the model that we'll use in a bit. I hope that as demand continues to rise for computing like these, costs will drop.[1]

Given the drawbacks of owning this computing today, and the hope that things may improve in the future, an effective counter option to the local running of these models is cloud-based platforms. If you are at a university, odds are you have access to some type of cloud computing, and you may be able to schedule jobs to run that require GPU for little or no cost when that computing is not being used by folks in STEM who brought in a grant to pay for it (Zaharia et al., 2016).

Using a Notebook Service That Has GPUs Available

If I were a graduate student today, I would use Google Colab, which provides various tiers of GPU resources for executing machine learning and deep learning research (Bisong, 2019). At the time of writing, the costs are quite reasonable ($0–$100 a month). It's built on top of Jupyter Notebook and allows for real-time collaboration and easy sharing of project files with your team. Google Colab offers a free tier with limited GPU access and the option to upgrade for more resources. Similar services include Kaggle Kernels, which also provide free and paid access to computational resources. For those with larger budgets or enterprise needs, cloud providers like AWS, Microsoft Azure, and Google Cloud offer powerful GPU instances that can be rented to run large models. These services provide a wide range of machine types and configurations to suit different computational requirements and budgets, allowing you to scale up or down as needed, but in general are very costly (Li et al., 2020).

You can leverage Google Colab for running larger models, like a 7B or 13B model, without worrying about the local hardware limitations. Right now, Google Colab only offers 40GB of video RAM maximum, and that isn't quite good enough to run 70B models (Bisong, 2019).

Using APIs

Another excellent option is to use OpenAI's API or Anthrophic's API which allow you to leverage the latest GPT models on a pay-as-you-go basis without worrying about the underlying hardware (OpenAI, n.d.).[2] OpenRouter takes this a step further and is an AI marketplace, where multiple vendors and their models can be swapped in a single parameter of code.[3] Using an

API approach provides scalability and ease of use, as you can make API calls to process your data and receive results over the internet.

ChatGPT is run by the parent company OpenAI. Under the hood, there is a simple API that anyone can use to programmatically ask questions. While it might be nice to use ChatGPT for one-off things, it's even better to build applications and scripts that handle daily digital chores. For instance, I use OpenAI's whisper API to automatically transcribe and summarize my lectures. From there, I generate a bank of quiz questions. I review those questions and edit them to my liking, and finally I generate lecture summaries and send those to students alongside the recordings.[4] The beauty of this lies in the fact that new quizzes and transcripts can be generated at the click of a button. The script runs, I programmatically ask GPT-4 to perform a series of summarization and question-generating tasks, and out spits a rough draft that I review. The last part, human review, is the most important (OpenAI, n.d.).

The major downside here is cost. OpenAI and others charge per token. This means that the cost can add up depending on the volume of text you are processing. I have spent $250 on one job in one day, just classifying a few thousand articles, but costs per token have dropped since then. Remember, each token typically represents a word or part of a word, so longer texts require more tokens, and thus the cost increases. If you are processing large amounts of data or require frequent interactions with the AI model, you may incur significant expenses.

Also, be aware that when you send data to an external API, you are transmitting potentially sensitive information to a third-party server. That's not something you should ever do with respondent data with sensitive data, (e.g., something you gained IRB approval for). Be careful with models you cannot directly control and treat the information you send as such.

Running Models Locally

While the days to come will almost certainly mean that more and more LLMs will run locally on the devices we own, like our laptops and phones, there is already a glimmer of immediate hope that some of these powerful models can run on high-end personal computers. Just recently I have had luck running LM Studio[5] on my Macbook Pro M2 Max from 2023. It runs smaller models at a slow, but usable speed. As Apple and other hardware creators begin to see the value of LLMs, I am sure newer devices will be able to do much more, but if you're set on keeping everything on your device, and you're not in a hurry, it's quickly becoming an option as well.

For the sake of this chapter, we're going to assume that you're going to either run your model locally on your expensive desktop GPU, or via a cloud notebook service like Google Colab.

Know Your Model

If you're considering labeling your data using open-source models, start with the Huggingface Model Library.[6] One of the standout qualities is just how user-friendly the library is. To discover models akin to the model we will use today, which is entitled "casperhansen/llama-3–8b-instruct-awq" on Hugging Face's model hub, search for key terms such as "Llama 3 8B" and "instruct." Refine your results to only include text generation. Examine the model cards for detailed insights, to verify you're using the model right. For instance, different models may ask you to format your prompt in differently. Also, be sure you're using the newest version. Leverage the "Similar Models" suggestions. Always ensure to review the documentation and licensing before using a new model in your project. Don't just use one because you found it in someone else's code, including mine. Models are constantly innovated, with newer, better models released daily.

Moreover, Huggingface actively encourages contributions from the community. Developers can train and share their own models on the platform, fostering a collaborative space for NLP enthusiasts. The library is also backed by an active open-source community, providing robust support and continuous updates based on the latest research. I find myself using models by the users TheBloke, or casperhansen. Why? Because these users focus on taking models that are very large and quantizing them into smaller files that can be run on cheaper GPUs (Wolf et al., 2020).

Each model in the library comes with detailed model cards, which include a description of the model, its performance metrics, the details of the training process, and the source code. This aids users in understanding the workings of the model and reproducing or building upon it. The documentation also provides instructions on how to customize and fine-tune the models based on specific requirements. Here's our model card: https://huggingface.co/casperhansen/llama-3-8b-instruct-awq.

From reading about this model, we know that the model was initially trained by Meta. LLaMA 3 is an open-source generative model initiative by the parent corporation of Facebook. Unfortunately, we don't know exactly what Facebook trained their algorithms on, but they do tell us that it's about 2 trillion tokens worth, so approximately half a billion words. They also let us know that they don't train on Meta product data (e.g., Facebook and Instagram posts) and that instead, the data comes from "publicly available sources" (Lewis et al., 2021).

Newer models like LLaMA 3 have, in my experience, closed the gap between OpenAI's GPT-4, with most leaderboards showing that the models can solve similar problems. In my experience, these models are quite similar, generally capable of talking and discussing basic problems and challenges. However, they also fail to understand very specific things, like specific mass

communication theories and the contemporary literature associated with them. Remember that all these models are unlikely to have been trained on paywalled journals due to copyright. They're at best going to have a surface-level understanding of detailed academic concepts as on Wikipedia, which is almost surely inside these models (Lewis et al., 2021).

Turning back to our model card, we see this is an "instruct" model because it was fine-tuned on open-source, long-context chats, and question-answer data. Fine-tuning is a crucial concept in machine learning that allows for pre-trained models to be effectively adjusted or "fine-tuned" to better serve a specific task. It works by taking a pre-trained model—that is a model that has already been trained on a large dataset and has developed an understanding of the dataset's features—and then training this model again, but this time on a new, task-specific dataset. The principal idea behind fine-tuning is to leverage and build on the already established knowledge and understanding of the pre-trained model, thereby enhancing its performance for the specific task (Houlsby et al., 2019).

In performing fine-tuning, the generative model is exposed to a smaller and usually less diverse dataset so it can specialize in specific features or characteristics. For instance, in the case of natural language processing, a pre-trained model like GPT-4 might be fine-tuned on a dataset of medical journal articles if the goal is to generate AI that can generate or understand medical text (Raffel et al., 2020).

Finally, we see that our model is 8B, meaning it's quite small as LLMs go. It will take only 5GB of GPU RAM to load and should work on the free version of Google Colab, provided you are connected to at least a T4 GPU. Note that the method we'll cover below dictates our model must be in Activation-aware Weight Quantization (AWQ) format (Hubara et al., 2017). Changing models is just a one-line parameter, so it's easy to try a few out to get one that works well for your problem, and you should spend time workshopping models before you label data.

Classifying Data Using Generative AI

In the previous chapter, we built an algorithm to address a contextual advertising problem with the aid of Keras and TensorFlow. We posed a classification task intending to identify news articles that delved into health and wellness, a particularly relevant category for certain advertisers (Chollet, 2017). In this chapter, we are about to broaden our horizons towards a goal that is more comprehensive and challenging. Our task now is to extend our classification algorithm capacity, not merely to discriminate health and wellness articles but to identify and categorize news articles based on the entire taxonomy of content provided by the Interactive Advertising Bureau (IAB).

IAB is an advertising business organization integrating over 650 leading media, marketing, and technology companies. Its principal role in ad tech is to develop industry standards, conduct research, and provide legal support for the online advertising industry. One of IAB's flagship contributions to this industry is the Content Taxonomy, a standardized classification taxonomy devised to streamline the process of content categorization (IAB, 2017). It represents a hierarchical arrangement of digital content into conceptual categories and sub-categories, making it an effective tool for contextual advertising.

By now, you should be thinking of this taxonomy as a set of codes in some master codebook. One frustration I have with the IAB is they don't provide robust conceptual definitions and examples needed to create a shared understanding of the content. Instead, they have ad tech companies classify the content according to their own definitions. This lack of intercoder reliability strikes me as problematic for advertisers. One ad tech company's contextual classifier is surely totally different from another's if they do not even share the same definitions.

We need to prompt our model to accurately map news articles to the appropriate categories in the IAB taxonomy. We need a generative model that is smart enough to understand, learn, and accurately predict a wide array of categories. Also, the model must navigate these numerous categories while still maintaining an acceptable level of precision and recall, ensuring that classification is not just exhaustive but also relevant and precise.

The core task in any generative AI query is to generate coherent and contextually viable text given some initial input. However, for our problem of content analysis using the IAB taxonomy, we need to transform this generated text into categorical labels that align with the specified IAB categories. At the heart of this transformation process is the premise that the generated response of the AI model will carry recognizable cues or keywords that can map onto the predefined categories in the IAB taxonomy. To accomplish this, we will implement a two-step process: model inference and response parsing into labels.

1. **Model inference**: In this step, we feed the news articles into the generative model. The format of the input includes the text summary of the article with some initial prompt that instructs the model about the required task. In our example, we will instruct the model to read the article content and identify the primary IAB category which the article fits into. The output of the model would then be a generated text snippet that explains the selected category and the reasoning behind the selection.
2. **Response parsing**: Once we obtain the generated response, we must parse this response and map it onto labels that correspond to categories

within the IAB taxonomy. This step requires careful prompt designing to ensure accurate mapping between the model's text output and the categorical labels. One simple approach can be to use keyword matching, where the presence of certain keywords in the model response selects a certain category.

Building a Generative Classifier

Importing and using a model from The Hugging Face model library is simply a matter of a few lines of code. The models can be directly downloaded and instantiated in the code using the "`from_pretrained`" function. This streamlined process allows even beginners in the field of NLP to implement complex models and experiment with different configurations (Wolf et al., 2020).

Switching between different models in the Huggingface library is equally straightforward. Given the uniformity of the API across different models, try out alternate models by changing the model's name in the "`from_pretrained`" function. Remember, exact loading instructions for each model can be found on its corresponding model card. Sometimes there are additional parameters you'll need to get it to load. First, we need to import the necessary libraries and initialize the LLM model.

To see how to initialize and run LLMs, I've included a working example in this chapter's notebook. We use vllm to serve the model. For the sake of this chapter, let's focus on the actual prompt that we use to get the LLM to respond:

```
{"role": "system", "content": "You are an expert at
contextual advertising and fully understand the Internet
Advertising Bureau's 3.0 taxonomy. You only respond with
classifications"},

{"role": "user", "content": Article:\n\n{document}\
n\n Return full article classification tree for the
article (e.g. 'Sports > Team Sports > Football > College
Football'). Return only the classification."}
```

This prompt both clearly and plainly explains the task and also provides an example output format for the LLM to respond with. The exact format of the prompt is entirely dependent on the model, which again can be found on the model card page on the model's Huggingface site, with different models having different formats. So be careful to review the model card to ensure you're generating the prompt right. When correctly formatted, the prompt instructs the model. If your model responds with odd strands of text, odds are it's not getting the prompt in the correct format.

Here, we're writing a straightforward and concise codebook to analyze the article and categorize it according to the IAB contextual video

categories. We don't need to describe each IAB category, because there are several documents on the open web describing the taxonomy. It's already in the generative AI's brains, as it were, and as such we just need to give the model enough context so it knows what it's classifying. This is likely not the case for obscure content analyses, so for your use case, you may need to explain more and/or provide a few examples.

Interactively and Iteratively Building a Codebook with Generative AI

In Chapter 5, we delved into the dataset released by the U.S. House of Representatives Permanent Select Committee on Intelligence, which contained advertisements orchestrated by the Russian Internet Research Agency (IRA) during the 2016 election. These ads were designed to incite outgroup incivility and sow discord among U.S. citizens by targeting various identity groups. The dataset, consisting of PDFs converted to text, provided a rich source for analysis but also posed a challenge due to the unstructured nature of the content and the difficulty in identifying specific instances of "othering" or incivility (House Permanent Select Committee on Intelligence, 2018).

We can use generative AI to explore what groups are in the data and eventually code the data as such. Remember, one of the key advantages of generative AI is its ability to understand context and generate responses that can help identify patterns or themes within the data. This is particularly useful when dealing with propaganda or incendiary content where the language used may be subtle or coded.

Generative AI can be employed to parse the IRA ads and extract mentions of specific outgroups without prior knowledge of which groups are being targeted. By crafting a generative query, we can instruct the AI to read through the ad content and identify any groups or entities that are being discussed in a potentially negative or divisive context.

Here's an example of how we might use a generative AI model like vLLM to extract such information:

```
# Prepare the generative query
query = "`[INST] <<SYS>>
You are a highly knowledgeable AI assistant with a deep
understanding of social dynamics and language nuances.
<</SYS>>
Your task is to analyze the given text from social media
advertisements and identify any groups or entities that
are being mentioned in a context that may suggest outgroup
incivility or 'othering'. Extract these mentions and
provide them in a structured JSON format with the following
properties: 'ad_id', 'text', and 'mentioned_groups'.
```

```
Advertisement Text:
{ad_text}
[/INST]"`

# Example ad text
ad_text = "Join us to stop the invasion of illegal immi-
grants who are taking our jobs and threatening our way of
life. America for Americans!"
```

The AI might return a response like:

```
{
"ad_id": "12345",
"text": "Join us to stop the invasion of illegal immigrants
who are taking our jobs and threatening our way of life.
America for Americans!",
"mentioned_groups": ["illegal immigrants"]
}
```

Note here that I am asking the generative AI to return JSON. In practice, this sounds great because this is a native data structure that Python can read. In reality, this can create many headaches, as generative AI models are infamous for *ever so slightly* producing outputs that don't conform to data structure standards. In these cases, your script will choke the over-malformed data. When in doubt, use a simpler text-searching method like the previous example to avoid these issues. Instead of using the JSON as-is, search inside of it.

Enhancing the Generative AI Query for Structured Extraction

To refine the generative AI's ability to extract specific groups of people according to their ethnicity, race, and country of origin, we can enhance the prompt to guide the AI towards a more structured and detailed output. This will help the AI to understand the task better and avoid the need to parse through coded or derogatory language used in the ads.

Here's an updated version of the generative query that includes instructions for extracting groups by ethnicity, race, and country of origin:

```
# Prepare the enhanced generative query
query = "`[INST] <<SYS>>
You are a highly knowledgeable AI assistant with a deep
understanding of social dynamics, ethnicity, race, and
country of origin.
<</SYS>>
Your task is to analyze the given text from social media
advertisements and identify any groups or entities that
```

```
are being mentioned in a context that may suggest outgroup
incivility or 'othering'. Specifically, extract mentions of
groups by their ethnicity, race, or country of origin. Pro-
vide the extracted information in a structured JSON format
with the following properties: 'ad_id', 'text', 'mentioned_
groups', and 'context_of_mention'.

Advertisement Text:
{ad_text}
[/INST]"`

# Example ad text
ad_text = "Join us to stop the invasion of illegal immi-
grants from [Country] who are taking our jobs and threaten-
ing our way of life. [Ethnicity or Race] do not belong in
our nation!"
```

The AI might return a structured response like:

```
{
"ad_id": "12345",
"text": "Join us to stop the invasion of illegal immigrants
from [Country] who are taking our jobs and threatening our
way of life. [Ethnicity or Race] do not belong in our na-
tion!",
"mentioned_groups": {
"ethnicity": "[Ethnicity]",
"race": "[Race]",
"country_of_origin": "[Country]"
},
"context_of_mention": "The advertisement uses divisive lan-
guage to create a sense of invasion and alienation towards
immigrants from a specific country and of a certain ethnic-
ity or race."
}
```

By providing a clearer definition of how the AI should return the outgroup information, we can avoid the laborious task of parsing through all the coded language in the ads. The AI is instructed not only to identify the groups but also to contextualize the mention, which helps with understanding the intent and impact of the advertisement.

If you find the AI returns a response where the context of mention is ambiguous or the identified groups are too broad, you may need to refine the prompt or codebook. Suppose the AI identifies "immigrants" as a mentioned group but does not specify the context in which they are mentioned. In that case, we might adjust our prompt to ask for more detailed information about the nature of the "othering" language used.

Here's how we might tweak the definition in our prompt to improve the AI's understanding:

```
# Revised part of the generative query
query = ″`. . .
Your task is to analyze the given text from social media
advertisements and identify any groups or entities that are
being mentioned in a context that may suggest outgroup inci-
vility or 'othering'. Specifically, extract mentions of groups
by their ethnicity, race, or country of origin, and describe
the context in which they are portrayed negatively or as a
threat. Avoid general terms and focus on specific language
that indicates targeted hostility or discrimination.
. . . ″`
```

By refining the prompt, we guide the AI to provide more granular and contextually relevant information, which can significantly enhance the quality of our codebook and subsequent analysis. Remember that being a "promptician" is an iterative process.

Only as Good as the Human in the Loop

The output from the generative AI may be good enough for your content analysis task if the problem is straightforward. However, we would be violating all the tenets of content analysis if we accepted the computer at its word. You must inspect the results manually and qualitatively. This inspection helps to ensure that the AI's interpretations align with our research objectives and the nuances of the content. In a generative AI content analysis, this is where I would expect researchers to spend most of their time. Honing your prompt so it returns the results expected. In areas of computer science, we call this a "human in the loop." As the social scientist and expert researcher, that must be you. You and your research team must play an active role in ensuring that any black-box tool you use works as intended.

This process could be as simple as reviewing a few hundred labels, generated by generative AI, and qualitatively adjusting the prompt and instructions until the AI performs as closely to humans as possible. At this point, we could calculate intercoder reliability measures like Cohen's Kappa, to Coder A (generative AI), and Coder B (you, the expert researcher). If we have reasonable agreement, we can rest easier knowing we've labeled the data in a way that generalizes. Alternatively, if we are focused on a single, relatively rare classification, such as whether an article is "misinformation," or not, the process could involve just reviewing the "positive" examples and correcting the labels where necessary.

That said, the transformative potential of generative AI in content analysis is just starting to be realized. We've seen how LLMs can parse complex and unstructured data, extract relevant information, and provide structured outputs that aid in our understanding of content. The ability of these models to generate contextually aware responses allows researchers to identify patterns and themes within data, even when the exact targets of analysis are not known beforehand.

As we close Chapter 6, recognize that generative AI is not just a tool for classification but a partner in the research process, capable of uncovering insights from data that might otherwise remain hidden. Its applications in content analysis will be vast and varied, and as we continue to harness its capabilities, we open up new possibilities for understanding and interpreting the complex world of human communication.

Notes

1 I hate to admit that I am too "I, me, mine" when it comes to owning computing, so I prefer to build my own machines and have my university host them for me. Which, they do! Go, buffs! I own several of the cards I've discussed so far and have hated trying to manage the CUDA software that goes along with them. For that, I could write a whole separate book.
2 More about OpenAI's API: https://platform.openai.com/docs/api-reference. More about Anthrophic's API: https://docs.anthropic.com/en/docs/intro-to-claude#api-reference
3 For more about OpenRouter: https://openrouter.ai/docs
4 Yes, I even used OpenAI's GPT-4 tool to generate the quiz and essay questions in the supplementary files that accompany this book. The use of this tool was intended to enhance idea generation and content creation. I have edited and reviewed the generated content.
5 The most capable model that I have had success with is LLaMA 3's 8B variant. Give it a try by downloading LM Studio here: https://lmstudio.ai
6 Access the model hub here: https://huggingface.co/models

References

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, *3(6)*, 1137–1155. https://doi.org/10.1162/153244303322533223

Bisong, E. (2019). *Building machine learning and deep learning models on Google Cloud Platform: A comprehensive guide for beginners*. Apress. https://doi.org/10.1007/978-1-4842-4470-8_7

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., . . . Amodei, D. (2020). *Language models are few-shot learners*. arXiv. https://doi.org/10.48550/arXiv.2005.14165

Chollet, F. (2017). *Deep learning with Python*. Manning Publications Co.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of deep bidirectional transformers for language understanding*. arXiv. https://doi.org/10.48550/arXiv.1810.04805

House Permanent Select Committee on Intelligence. (2018). *Exposing Russia's effort to sow discord online: The Internet Research Agency and advertisements*. https://intelligence.house.gov/social-media-content/

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., . . . & Gelly, S. (2019). Parameter-efficient transfer learning for NLP. *Proceedings of the 36th International Conference on Machine Learning, 97*, 2790–2799.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., & Bengio, Y. (2017). Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, *18*(1), 6869–6898.

Interactive Advertising Bureau. (2017). *IAB Content Taxonomy*. https://www.iab.com/guidelines/content-taxonomy/

Krippendorff, K. (2022). *Content analysis: An introduction to its methodology*. Sage.

Lewis, M., Zettlemoyer, L., Stoyanov, V., & Riedel, S. (2021). *Retrieval-augmented generation for knowledge-intensive NLP tasks*. arXiv. https://doi.org/10.48550/arXiv.2107.07566.

Li, M., Andersen, D. G., Park, J. W., Smola, A. J., Ahmed, A., Josifovski, V., Long, J., Shekita, E. J., & Su, B.-Y. (2020). Scaling distributed machine learning with the parameter server. *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation*, *2014*, 583–598.

NVIDIA. (2020). *NVIDIA A100 Tensor Core GPU Architecture*. https://www.nvidia.com/en-us/data-center/a100/

OpenAI. (n.d.). *OpenAI API*. https://platform.openai.com/

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, *1*(8), 9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li., W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, *21*(140), 1–67.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.) *Advances in neural information processing systems*, *30* (pp. 5999–6009). Neural Information Processing Systems Foundation.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Xu, C., Le Scao, T., Gugger, S., & Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. In Q. Liu & D. Schlangen (Eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 38–45).

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., & Stoica, A. (2016). Apache Spark: A unified engine for big data processing. *Communications of the ACM*, *59*(11), 56–65. https://doi.org/10.1145/2934664

Chapter 7

Topic Modeling as a Lens for Discovery

When Should We Use Topic Modeling in Content Analysis?

As I was coming up as a doctoral student and young professor, about a decade ago, there was a bit of hype in the mass communication literature about "big data."[1] I'm glad the hype has died down a bit as we all know the saying "bigger isn't always better." As we discussed in earlier chapters, when we're looking at "big data," that doesn't give us carte blanche to apply black boxes to the data and call it a day. Algorithms can help us make sense of the data by reporting commonalities—the words tend to cluster and appear together in documents across a corpus. This initial understanding is important if our research is to be truly inductive and receptive. At the core of the social sciences lies the imperative to listen—to allow the data to speak and to induct themes with minimal presuppositions. This endeavor aligns seamlessly with the principles of topic modeling. Through topic modeling, we invite the data to unveil its hidden narratives and themes in their most unadulterated forms.

In the past examples we've looked at, we have known from the start what we've been looking for, incivility, political talk, items from a contextual advertising taxonomy, and so on. When items are known they are somewhat easy to extract from the data. There are only so many words you can use when talking about politics, and only so many insults that can be levied. However, what if we have a collection of data we know little about? For instance, if we have a collection of Reddit posts from a collection of subreddits, we may know a bit about the interests of those subreddits, but if we had to guess at a list of common topics, we'd have to manually review the subreddit data for hours before we even begin to make sense of the most recent posts on these subreddits, and that would just scratch the surface if we wanted to look at Reddit across a long period.

At the heart of unsupervised machine learning lies the concept of discovery without defined parameters—sifting through unstructured data to detect

DOI: 10.4324/9781003514237-7

patterns, groupings, or structures without explicit instruction. This form of machine learning is invaluable when the researcher does not have predetermined categories or labels, essentially stepping into a room with no prior knowledge of what's inside and allowing the contents to reveal their own story.

This statistical method discerns clusters or "topics" defined by their distinctive vocabularies. Each topic emerges through an iterative process of comparison and correlation, resembling themes that are consistently reflected across numerous documents within a dataset.

Topic models, particularly Latent Dirichlet Allocation (LDA) introduced by Blei, Ng, and Jordan in 2003, initially gained traction within the domain of information retrieval as a sophisticated method for organizing and assimilating large archives of unstructured text. Before LDA, information retrieval systems often relied on simpler techniques such as keyword matching and simple text queries, which could not capture the nuanced relationships between words and documents. Topic models transcended these limitations by extracting latent thematic structures from text, thus allowing for a richer, more contextual understanding of content.

Today, topic models have built on their early promise and have been adapted for a vast array of applications, ranging from enhancing recommendation systems that consider user interests and content semantics to analyzing social media trends for public sentiment and cultural analytics. By revealing the underlying themes within massive datasets, topic models are currently used in interdisciplinary research to unveil insights in fields such as digital humanities, computational social sciences, and bioinformatics. Moreover, topic modeling has grown in sophistication to handle multimodal data sets, including a mix of textual, visual, and audio information, thus providing a multi-dimensional perspective in the automated analysis of large, diverse datasets.

The utility of topic modeling is particularly prominent in mass communication research where scholars endeavor to grasp the complexities of public discourse as mirrored in online platforms. For instance, in the analysis of news website comment sections, topic modeling can parse out underlying topics of public concern or contention, such as climate change or immigration, without the researchers imposing their own interpretive frameworks. Similarly, within the realm of social media analytics, topic modeling can illuminate trending conversations on social media platforms around global events, providing a pulse on collective sentiment and attention dynamically over time. However, there are many times when topic models are not the best method to use.

Latent Dirichlet Allocation (LDA) Topic Modeling and Its Limitations

It is my educated guess that the most popular topic modeling approach used in mass communication is LDA, and for many years it was the best solution to topic modeling on big data. The mathematical equation LDA uses is

quite simple and intuitive. When authors like you or I write, we have certain topics in mind, and we choose words that are relevant to these topics. LDA operates under the assumption that each document is a mixture of various topics, and each topic is characterized by a set of words that frequently occur together. The "Latent" part of LDA suggests that these topics are hidden or not directly observable. "Dirichlet" is a reference to the type of statistical distribution used to model the variability in the data, which in simpler terms helps us account for the fact that not all topics are represented equally across documents. Some topics might be more prevalent, while others are less so. "Allocation" refers to how words are distributed across these topics.

The true power of LDA lies in its iterative nature. With each pass through the data, LDA adjusts its guess of where the words should be allocated. It does this by evaluating two key probabilities: first, the probability that a document would contain a particular topic (given the current assignment of words to topics across all documents); and second, the probability that a topic would contain a particular word (given the current assignment of topics to this and all other words in the documents).

Consider a corpus of articles from various news sections—politics, sports, entertainment, and so on. LDA doesn't know these categories; it only sees words. In the beginning, it might place the word "ballot" in the same topic as "football" simply because it doesn't know any better. However, as it iterates, it notices patterns: "ballot" often appears with words like "election" and "vote," while "football" appears with "game" and "team." LDA uses these co-occurrence patterns to adjust its guesses.

The algorithm asks, "Given that 'ballot' appears in this document, how likely is it that the document is about politics?" and "Given that this document is about politics, how likely is it to contain the word 'ballot'?" By repeatedly asking these questions for every word in every document, LDA refines its topics. Words that frequently appear together move towards the same topic, while those that don't drift apart. In essence, LDA is a method that mirrors the inductive reasoning process, starting with broad strokes and gradually locking in on the finer details, allowing us to uncover the latent structures that shape the content of our textual data.

LDA and all topic modeling approaches don't give you the final answer but rather a set of topics that are statistically likely. It's then up to you, the researcher, to interpret these topics. You might find that some topics are exactly what you expected, while others are surprising. This is where your expertise as a social scientist comes into play, as you interpret and give meaning to these statistical findings, weaving them into the narrative of your research.

LDA often overlooks the nuanced and overlapping nature of language. For example, the word "bank" could be associated with topics of finance, river ecology, or even a collection of items, depending on the context. LDA lacks the sophistication to discern these distinctions on its own because it has

none of the word embeddings or natural language context that we've discussed with deep learning. LDA doesn't handle polysemy well. When a single word has multiple meanings, LDA can't disambiguate. LDA's statistical nature struggles to understand context at the complexity level that human language demands. The words themselves don't carry semantic meaning in the model—only their distribution does, which can lead to erroneous or overly broad topic assignments. In practice, despite spending hours fitting topic models with fit statistics, word distributions of topic models often need to be cleaned up by humans by removing words that were spuriously correlated.

Over the years I have tried to fit various LDA implementations on various media data sets. I have been consistently reminded that topic models are not refined classifiers like those employed in deep-learning architectures, such as we discussed with TensorFlow. Remember, supervised machine-learning classifiers are trained on labeled datasets and become precise scalpels. They learn representations of data at multiple levels of abstraction, allowing them to capture complex patterns within the text, including context and word order—something LDA is inherently unable to do.

Imagine you are trying to understand the topics discussed in a collection of documents, let's say a set of news articles. You want to know which words are important for each topic, just like in regression analysis where you find the important predictors for an outcome variable.

Topic models, such as Latent Dirichlet Allocation (LDA), are tools that help us figure out which words are most important to different topics within a large collection of texts. LDA works by itself to determine how likely each word is to be part of a specific topic. This likelihood is based on how often words appear in relation to each other throughout all the texts we're studying. Think of these likelihoods as scores that tell us how relevant a word is to a topic—the higher the score, the more relevant the word. It's like regression coefficients: high scores can tell us that there's a strong connection between two things we're interested in.

It's crucial to understand that topic models, while having similarities to regression and bag-of-words classifiers, are designed for exploratory data analysis rather than prediction. Consequently, the "coefficients" (word-topic probabilities) of a topic model do not serve the same function as those in more deterministic models. They don't predict an outcome but rather describe the underlying structure of the data set, which is the distribution of words across inferred topics.

The inherent logic of advanced machine-learning techniques like TensorFlow-based deep-learning models is that they excel in document-level classification tasks. This specific superiority stems from their ability to capture complex patterns, semantic relationships, and nuanced contexts within the text-critical factors that distinguish one document class from another.

Deep-learning networks go far beyond the surface-level patterns identified through topic modeling by leveraging multiple layers of abstraction.

TensorFlow and deep-learning architectures are also adept at handling the idiosyncrasies of human language, including context, order, and syntactical nuances, which are often pivotal for accurate classification. For example, in sentiment analysis, the phrase "not good" within a review carries a distinctly negative connotation—a subtlety that deep-learning models can discern but which might elude a basic topic model like LDA unless it included bigrams as features. Furthermore, with the ability to learn from annotated data, these models can adjust weightings across complex neural networks to optimize classification performance, a feat unattainable with the rigid, generative assumptions of LDA.

Topic Modeling as a Starting Point to Excellent Training Data

I do not intend to downplay the value of topic modeling in the inductive research process. As a first step, topic modeling can quickly sift through voluminous data, providing a macro-level view that identifies clusters of content for closer inspection. It can guide researchers by unveiling thematic signposts and illuminating areas of interest that might benefit from further, more granular analysis.

Nevertheless, such an initial step should not be the endpoint, unless the study aims to explore something unknown. For instance, in this chapter, we'll unpack how I used topic modeling to explore commonly deleted posts on Reddit. I argue that it's a valid research tool because the commonly deleted topics have not yet been uncovered and discussed in the literature.

However, we didn't exactly do a content analysis of the data. I didn't accurately classify the prevalence of each topic, nor did I take these topics as specific measures of scientific concepts. When we have these objectives, a supervised machine-learning approach, whether it be deep or linear, is the best way to classify the data. I often think of what my colleague once said, summarizing what we do as researchers in mass communication, "really, we are just professional bean counters," and if that is our aim in a study, we should do it with precise tools. However, if our aim is to act as archeologists of media data, then topic models can certainly uncover unknown commonalities.

Preliminary insights from topic modeling can inform and fine-tune deep learning models. The broad thematic landscapes revealed by topic modeling can expedite the preparation of labeled datasets for supervised learning or guide researchers in crafting more targeted neural network architectures. Imagine you wanted to separate political talk from other types of discourse on Reddit, to further study who shares it, what types of political talk get more engagement, and so on. We could use a database like Pushshift.io to

pull a list of submissions from popular subreddits.[2] This might entail sampling a significant and diverse array of submissions across a range of potentially politically active subreddits over a defined period, say a year or two.

If you were to randomly sample posts for whether they were political or not, you would find somewhere around one out of ten posts would be political in nature. As a content analyst, there are two downsides here. First, you're wasting your time reading a lot of irrelevant (non-political) content, and you're also creating an imbalanced training set for a deep learning algorithm—something that we're going to have to fix by essentially throwing out a bunch of negatively classified documents in the training process, anyway. In this case, you need the needles in the haystack. Topic modeling can help you find them. Remember, topic models are for uncovering the broad thematic outlines within the collected data. This step helps identify which documents are likely to contain political conversation without yet knowing any specifics. We can then pull documents for topics that appear to have political words in them and scrutinize them with our own eyeballs.

From here it's easy to scan the topics produced and filter to only topics that are likely to contain political talk. In preparing an analytical sample, you can include likely documents from these topics. I encourage you to also select a random sample of 10% or so of unfiltered data to ensure you're not missing something. Even if your unfiltered sample is much higher than this, by selecting from likely topics, you will drastically increase the needles in your haystack. This will create a much better training set for your supervised machine-learning algorithm with fewer human annotations overall.

Provided you adhere to the principles of content analysis we discussed at the beginning of the book, and your data is labeled validly and consistently, you now have what you need for a great supervised machine-learning algorithm. Recall that supervised machine-learning algorithms have clear training, testing, and validation processes that ensure the classifier is conforming to the documents, and this is what we are gaining by going this route over topic modeling.

In this chapter, we'll cover how to build a topic model with BERTopic over traditional LDA because it utilizes contextual embeddings to capture the semantic meaning of words. This means it surpasses the constraint of word-frequency patterns that LDA relies on, accommodating the context in which words appear. BERTopic is gaining popularity over LDA not only because of its semantic awareness and flexibility but also due to its compatibility with large datasets and rigorous real-world applications like this one. It leverages the same BERT we discussed in Chapter 5. Recall that BERT's prowess lies in its transformation of traditional bag-of-words approaches into something significantly more dynamic, capturing the elusive essence of language: its context-dependent meaning.

Building a BERTopic Model on Reddit Posts

Let's walk through a topic model I recently published. In online communities like Reddit, posts are constantly in flux, subject to the editing whims of users and the discerning actions of moderators. We were especially captivated by the questions lurking within the deletions: Were there discernible patterns that could be gleaned from the removed content? Did specific themes emerge that pointed to broader trends in community norms or policy enforcement? These were not questions of a binary nature, but rather of a thematic breadth—a perfect fit for the strengths of topic modeling. Faced with the enormity of Reddit's data and the intricate puzzle of content moderation, we elected to employ topic modeling as our investigative tool.

We wanted to draw a representative sample of active subreddits. To do so, we collected the top 100 "safe for work" subreddits as calculated by redditlist.com. This list broadly represented subreddits that were known to have relatively high levels of recent activity. To avoid seasonality issues, we sampled a two-year period from January 1, 2021, to December 31, 2022. With these research questions in hand, we trained our topic model on a set of posts that were deleted, as evident from the metadata inside of the post.

We chose BERTopic for its ability to isolate topics within large text corpora by leveraging contextual embeddings, providing a depth of understanding beyond the reach of mere frequency counts that traditional models like LDA rely on. As we employed BERTopic to unravel the hidden threads within the deletions, we hoped to illuminate the subtle contours of topics that recurred within the removed content. The idea was not only to catalog these topics but to engage with them, to understand the implicit rules and spoken silences of the Reddit ecosystem.

The code provided in this chapter's notebook creates a "`representation_model`" that is "`KeyBERTInspired`" (Grootendorst, 2020). The term "KeyBERTInspired" suggests that the representation model is influenced by KeyBERT, a keyword extraction algorithm that leverages the BERT language model. Recall that BERT is designed to understand the context of words in a sentence by looking at the words that come before and after, rather than in isolation (Devlin et al., 2019). We then initialize a "`BERTopic`" object, passing our representation model as an argument to tailor the topic discovery process. Once our model is set up, we feed it our "`posts`" list. The "`fit_transform`" method trains the model on our data and assigns each document in "`posts`" to its most probable topic, returning two arrays: "`topics`" and "`probabilities`." The former gives us the topic for each document, while the latter gives us the probability of each document belonging to its assigned topic—a measure of how confidently the model believes the assignment is correct (Blei et al., 2003).

This process, as coded, embodies the very essence of topic modeling—it is the critical algorithmic step where patterns are detected, themes emerge, and the veil of large-scale data complexity is lifted. The resulting topics form a base on which further qualitative and quantitative analyses can build, painting a clearer picture of the vast landscape of Reddit's moderated content.

Interpreting Topics with Generative AI

Topic models like BERTopic are adept at grouping words statistically, but they do not inherently provide insights into what these groups represent in human terms. It's akin to presenting someone with a bag of puzzle pieces without the picture on the box—recognizably part of some whole but lacking coherent meaning. To remedy this, a qualitative review of the documents will unlock meaning for the researcher, but again when we consider very large datasets, manually reading a representative sample of documents from each topic in a dataset may be too arduous. In these situations, generative AI can reconstruct the image on the box, synthesizing a coherent picture from the disparate pieces.

Most topic models are entitled with the top *n* number of words that represent that topic. While we may be able to see a set of words and infer the meaning in a qualitative sense—it is easy to infer what topic "musk," "twitter," and "advertisers" are about for instance—other topics may be broader and therefore obtuse to an immediate meaning.

As I've mentioned, generative AI is good at summarizing long things, like video transcripts, textbooks, and so on. As such, in our study, we fed large samples of Reddit posts from each topic to the generative AI model GPT-4 and tasked it with interpreting the topics identified by BERTopic, in a process reminiscent of a qualitative coding session. The generative model's capability to generate human-like text was directed to produce descriptions and narrations that encapsulate the essence of each topic, which you can find in Appendix 1. I include them because I think they show just how well these models can understand and summarize large corpora.

```
# Print major topics and their associated words
topic_words = topic_model.get_topic_info()
topic_terms = topic_words['Representation']
topic_docs = topic_words['Representative_Docs']

all_topic_notes = {}
for i, a_topic in enumerate(topic_terms):
topic_text = str("%s %s" % (a_topic, topic_docs[i]))
response = gpt4_query(topic_text, 0, 50)
responses = response.split(<TAB>")
print("--------------------------")
for a_response in responses:
print(a_response)
```

```
all_topic_notes[i] = responses
```

In the code, "`get_topic_info`" extracts the words and documents associated with each topic generated by BERTopic. These are used to construct prompts for the generative AI model. The "`for`" loop iterates over these topics and documents, crafting a string that encapsulates both. This string is then passed to GPT-4, which processes the information, provides a human-readable summary of the topic's contents, and posits reasons why such topics may frequently be deleted by users or moderators.

These AI-provided descriptions and speculations fundamentally bridge the gap between statistical analysis and human understanding. The AI's output is not just a compilation of words but a crafted narration that captures potential community norms, editorial processes, and cultural contexts—an interpretation that is easy for humans to engage with and scrutinize further.

Moreover, GPT-4's ability to work with complex and nuanced language means that it can handle the subtleties and ambiguities inherent in human discourse. Its output provides a rich qualitative understanding that not only identifies what the topics are about but also explores and provides possible reasons as to the "why"—why these posts may have been deleted or moderated.

By integrating generative AI into the analytical process, we transformed the quantitative data from BERTopic into narratives and interpretations that could then be reviewed, questioned, or corroborated by human researchers. This symbiosis of machine learning's computational power and generative AI's linguistic agility illustrates a powerful new paradigm for qualitative data analysis in the digital age.

To produce a similar workflow, you'll need a set of OpenAI API keys, which is necessary to authenticate and authorize interactions with OpenAI's services.[3]

```
openai.api_key = "your-api-key"
```

Next, we define a function "`gpt4_query`" that takes a string of text "`news_article_text`" and a temperature setting for randomness "`a_temp`." In content analysis, setting the temperature parameter to 0 when using LLMs ensures that the output is deterministic and consistent. This is crucial for maintaining the reliability and validity of the analysis, as it eliminates the variability and unpredictability in the responses that a higher temperature setting would introduce. By using a temperature of 0, researchers can be more confident that the AI's output is based solely on the most likely response given the input data, without the influence of randomness that could skew the analysis. We also specify a maximum token limit "`max_tokens`" to control the length of the generated content, as we don't want the AI to ramble on forever.

```
def gpt4_query(news_article_text, a_temp, max_tokens):
# Instructions and codebook designed for guidance of the AI
query = [
{"role": "system", "content": "You are an expert qualita-
tive researcher. "},
{"role": "user", "content": "%s" % CODEBOOK},
{"role": "user", "content": "%s" % news_article_text},
{"role": "user", "content": "%s" % INSTRUCTIONS},
]

# Attempt to retrieve a response from GPT-4 with retry log-
ic for potential rate limits or API errors
for attempt in range(20):
try:
response = openai.ChatCompletion.create(model="gpt-4",
temperature=a_temp, messages=query, max_tokens=max_tokens)
response_text = response['choices'][0]['message']['content']
return response_text
except (openai.error.RateLimitError, openai.error.APIError)
as e:
if attempt < 19:
print(f"{type(e).__name__}: {e}. Retrying in {2 ** attempt}
seconds . . . ")
time.sleep(2 ** attempt)
else:
print(f"{type(e).__name__}: Maximum retries reached. Giving
up.")
raise
```

The "`gpt4_query`" function includes a query that contains a set of instructions for the AI; this is where you need to spend the most of your time. This is our codebook, and it will be followed quite literally, for better or worse. The query we are creating here is essentially modeled as a chat session that you could have with ChatGPT, LLaMA 3, or any of the models available on the Huggingface model library. Note the messages that specify the role (either "system" or "user") and associated content. The "system" message is used to set the context, indicating that the AI should act as "an expert qualitative researcher." The "user" messages include the provided "`CODEBOOK`" for context, the text to analyze "`news_article_text`", and the "`INSTRUCTIONS`" to guide the AI's response.

In the main loop, the code builds up an "`all_topic_notes`" dictionary to store the AI-generated responses:

```
all_topic_notes = {}
for i, a_topic in enumerate(topic_terms):
topic_text = str("%s %s"% (a_topic, topic_docs[i]))
response = gpt4_query(topic_text, 0, 50)
```

```
responses = response.split("<TAB>")
print("----------------------------")
for a_response in responses:
print(a_response)
all_topic_notes[i] = responses
```

The loop iterates over the topics, sending each as input to "`gpt4_query`" and processes the output by splitting it with "`<TAB>`." This is because the GPT-4 response structure is expected to follow a format where the summary is followed by "`<TAB>`" and then a speculated reason for content deletion on Reddit. These results are printed and stored for further use.

Finally, once all the responses are gathered and stored, they are later used to visualize topics and possibly refine the results:

```
pickle.dump(posts, open("%s/all_topic_notes.p" % WORKING_
DIR, 'wb'))
# Visualization functions are called to plot the topics in
different ways
topic_model.visualize_topics()
topic_model.visualize_hierarchy(top_n_topics=50)
```

This part of the code saves the "`all_topic_notes`" to a pickle file and then calls visualization methods provided by BERTopic to scrutinize the model's topic clusters visually.

Appendix 1 provides a fascinating lens through which we can view the often invisible forces that shape discourse within the diverse ecosystem of Reddit. The generative topic summaries muster a narrative far richer than what one might extract from a barren list of words typically associated with each topic. Words without context serve as mere signposts, indicating potential directions but not the nature of the journey taken, or the destination reached. For instance, a raw collection of words drawn from topics related to "financial and economic topics"—cryptocurrency, stocks, investment—lacks the dimensionality to convey the underlying reason for a post's deletion, such as the propagation of questionable investment advice or the potential harm from misinformation.

On the other hand, the generative topic summaries incisively cut to the core of the matter. They delineate not only the essence of the deleted content but also the implicit societal, communal, and normative considerations underpinning such action. For instance, when we read that a post about "sensitive or distressing content" is deleted, we understand it's not merely the topic that's at issue, but the potential emotional impact and spread of fear that's flagged as problematic (Chancellor et al., 2016).

Consider the contrast between the unembellished topic words "meme, joke, humor" and the enriched understanding provided by the summary explaining these are often deleted due to their low-quality or non-contributive

nature to meaningful discussions. The AI-transformed topic summary might read: "Removed Content: Frivolous Humor—Content generating more noise than insight, falling short of communal aspirations for substantive dialogue."

Furthermore, the AI's qualitative summaries render otherwise abstract concepts—such as "privacy concerns" or "inappropriate content"—into tangible illustrations of community dynamics. A human analyst might interpret a cluster of words like "HIV, transmission, sexual content" with clinical detachment, but when it's captured in the generative summary, it reveals concerns about potential misinformation and user safety that resonate on a more human level.

These narrative-rich insights provide an accessible foundation for further investigation. They serve as the qualitative bedrock from which one could, for instance, draft precise guidelines for content moderation or ethical considerations in community management. In comparison to the raw topic words, the generative summaries offer a bird's-eye view of the social and emotional landscapes that online communities navigate, emphasizing the importance of interpreting data not just scientifically, but also humanely and ethically.

Summing Up Topic Modeling's Role in Content Analysis

Topic modeling emerges as both a lantern illuminating the vast and often chaotic expanse of media data and a tool whose efficacy is contingent on the contours of the research landscape. It's a quintessential example of the power and limitations inherent in unsupervised machine-learning methods—capable of drawing out patterns from the noise but constrained by the very assumptions that underpin its statistical foundations (Blei et al., 2003; Griffiths & Steyvers, 2004).

The very nature of topic modeling—that of a blunt instrument rather than a scalpel—can also be a limitation. The topics generated, while directionally valuable, may lack the precision required for nuanced analysis or the construction of predictive models. For instance, in settings where the objective is to make fine-grained distinctions between document classes, such as identifying sentiment or specific content types like political rhetoric, the unsupervised nature of topic modeling usually falls short. Here, the rough thematic brushstrokes painted by LDA or even BERTopic, in my experience, fail to capture the subtleties needed for high-stakes decisions, such as nuanced content recommendations or the identification of harmful content meriting removal (DiMaggio et al., 2013).

Moreover, topic modeling, for all its virtues, cannot single-handedly bridge the gap between detecting topics and understanding them. The interpretative leap from a cluster of words to a thematic narrative requires a qualitative touch that must be conducted with humans in the loop, and those humans must provide qualitative scrutiny and nuanced contexts.

Thus, as an initial step in a larger research design, topic modeling can demystify large-scale data, guiding subsequent, more refined analyses. It is a tool for researchers to wield when the journey begins with questions rather than answers when the paths through the textual woods are numerous and the possibility of discovery is boundless. Yet, as the journey advances toward the specific and the defined, researchers need to step beyond the threshold that topic modeling provides and delve into more targeted and supervised machine-learning techniques.

Notes

1 As I put forward earlier in this book, "big data," in terms of a content analysis, is a corpus of data that is too big for a team of researchers to review a representative sample of.
2 Due to changes in reddit's data policies, Pushshift is unavailable for academic research as of March, 2023, but archives of data persist: https://academictorrents.com/details/89d24ff9d5fbc1efcdaf9d7689d72b7548f699fc
3 To get started with OpenAI and obtain an API key, you'll first need to create an account on the OpenAI website. Visit https://platform.openai.com/ and sign up for an account by providing your email address and creating a password. Once your account is set up, you can access the API section from your dashboard, where you can apply for an API key. OpenAI will guide you through the process, which may include agreeing to terms of service and possibly providing payment information if you're subscribing to a paid tier of service. After completing these steps, you'll be issued an API key, which you can use to authenticate your applications and start interacting with OpenAI's suite of AI tools and models.

References

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022. http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

Chancellor, S., Pater, J. A., Clear, T., Gilbert, E., & De Choudhury, M. (2016). #thyghgapp: Instagram content moderation and lexical variation in pro-eating disorder communities. *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, 1201–1213. https://doi.org/10.1145/2818048.2819963

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of deep bidirectional transformers for language understanding*. arXiv. https://doi.org/10.48550/arXiv.1810.04805

DiMaggio, P., Nag, M., & Blei, D. (2013). Exploiting affinities between topic modeling and the sociological perspective on culture: Application to newspaper coverage of U.S. government arts funding. *Poetics*, *41*(6), 570–606. https://doi.org/10.1016/j.poetic.2013.08.004.

Grootendorst, M. (2020). KeyBERT: Minimal keyword extraction with BERT. *The Journal of Open Source Software*, *5*(52), 2455<AQ: page span?>.

Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences*, *101*(Suppl 1), 5228–5235. https://doi.org/10.1073/pnas.0307752101

Chapter 8

Extending Deep Learning to Image Content Analysis

In mass-media research, the profound influence of images on public consciousness and discourse is increasingly acknowledged. Moreover, as more digitally capturable experiences like the metaverse come to life, there will be more opportunities to capture and annotate these experiences. As potent conveyors of meaning, sentiment, and ideology, images not only transcend linguistic and cultural barriers but also play a pivotal role in shaping narratives and perceptions (Messaris, 1997).

As we can analyze and label text using machines, so can we with images. Really, that's the only other medium I can think of that we need to analyze from a mass-media perspective. What about video? Videos are but many images, and while there are new multi-modal models that handle video directly, they still are yet to be as accessible as image AI.

What about audio? Audio can be transcribed to text using free tools like OpenAI's Whisper.[1] While currently, we may lose things like music, and the context in which the audio is occurring (e.g., a live concert, a boardroom, at a park), we can easily extract the information from audio and make it text.

Therefore to close this book I have chosen to explore how vision AI can be harnessed to decode the complex interplay of visual elements, offering researchers a powerful tool to quantify and interpret the subtle nuances that influence public opinion, media effects, and consumer behavior (Zhao et al., 2019).

Neural Networks and Images

Shockingly, I haven't had to explicitly talk about neural networks yet in this book. At the heart of supervised deep learning lies the concept of neural networks, which are computational models designed to recognize patterns in data through a process that mimics the way human brains operate (LeCun et al., 2015). Neural networks consist of layers of interconnected nodes, or neurons, each of which processes input data and contributes to the network's

DOI: 10.4324/9781003514237-8

output. Deep learning is almost always preferred when trying to classify images, because, unlike text, image features are hard to extract. While we can easily interpret the meaning of words, the meanings behind the pixels and bytes of an image, or a series of images (e.g., a video), are much more difficult. In a deep-learning problem designed to classify images, the first layer, known as the input layer, receives the raw data, which, in the context of image analysis, typically comprises pixel values of images. Subsequent hidden layers extract features from this data, and the final layer, or the output layer, provides the results of the analysis, such as the classification of images into different categories based on their content (Goodfellow et al., 2016).

As we discussed with TensorFlow and Keras earlier, in deep learning the loss function serves as a measure of how well the neural network is performing. It calculates the difference between the network's predictions and the actual data, guiding the adjustment of the network's weights through a process known as backpropagation (Rumelhart et al., 1986). By iteratively minimizing the loss function during training, the neural network learns to improve its predictions, becoming more adept at analyzing and interpreting image data.

Datasets for Image Content Analysis

Global Database of Events, Language, and Tone (GDELT) stands out to me as an invaluable resource for analyzing image content from different news media sources around the web (Leetaru & Schrodt, 2013). Its automated data-crawling techniques are very useful to mass-communication researchers. Due to copyright restrictions. it cannot give researchers the text of articles; however, it does provide all the hyperlinks to images it finds in news stories.

That means GDELT is a historical archive of news article images.[2] I've been using it to analyze and capture news-media images for social science research. This collection is particularly useful for researchers interested in the visual framing of news and the portrayal of events across different cultures and media outlets. The visual data available in GDELT can be used to track the evolution of news stories, identify patterns in media coverage, and analyze the impact of visual representations on public perception (Kwak & An, 2016). For instance, by applying image recognition algorithms to this dataset, one can quantitatively assess the prevalence of certain visual themes or subjects in news media over time.

The Internet Archive serves as a digital library, offering a historical archive of web pages and their associated media, including images. For researchers, this represents a treasure trove of historical web imagery that can be used to study the evolution of internet culture, web design trends, and the role of imagery in online communication. By examining these historical images, scholars can gain insights into the visual dimensions

of past online discourses, the aesthetics of information presentation, and the shifts in cultural preferences over the years. The temporal depth of the Internet Archive's collection allows for longitudinal studies, which are essential for understanding changes in visual communication strategies on the web (Brügger, 2018).

Contemporary social media imagery, on the other hand, provides a window into current societal values, interests, and behaviors. Platforms like Pinterest and others that offer open APIs enable researchers to access a wide array of user-generated images that reflect contemporary culture. These images can be analyzed to discern patterns in visual self-representation, branding strategies, and the diffusion of visual memes. Social media platforms are particularly relevant for studying the role of images in peer-to-peer communication and the viral spread of visual content. By leveraging these APIs, researchers can create datasets that are not only large in scale but also diverse in representation, encompassing a multitude of user demographics and cultural backgrounds (Highfield & Leaver, 2016).

Kaggle, an online community of data scientists and machine-learning practitioners, offers a plethora of datasets that are primed for immediate use in research. These datasets often include a wide range of image categories, such as those found in news articles or disseminated via social media platforms. The inherent value in utilizing a dataset from Kaggle lies in its readiness for analysis, often accompanied by pre-assigned labels that delineate the content of the images across various categories. This pre-processing of data is a significant boon for researchers, as it circumvents the labor-intensive process of manual labeling, thereby accelerating the research workflow. However, when selecting a dataset from Kaggle for image content analysis, it is also essential to consider the diversity and representativeness of the image samples. Mass communication is a field that thrives on the examination of varied perspectives and contexts. Hence, a dataset that encompasses a broad spectrum of image sources, geographical locations, and cultural contexts will likely yield more generalizable and insightful findings (Zook et al., 2017).

High-Level Python Packages for Image Analysis

Once you have a large collection of images, it is indeed possible to use deep-learning models, particularly TensorFlow's Keras API[3] and PyTorch's torchvision to label images. Both of these abstract much of the complexity inherent in model building and training. Keras simplifies tasks such as the construction of neural layers, activation functions, and optimizers, allowing for the rapid prototyping of computational models. Moreover, its comprehensive set of pre-trained models and weights can be leveraged for applications such as classification, segmentation, and feature extraction in image analysis tasks, significantly reducing the time and effort required for model development and training (Chollet, 2018).

Parallel to TensorFlow's Keras is PyTorch's torchvision, a package that consists of popular datasets, model architectures, and common image transformations for computer vision. PyTorch, developed by Facebook's AI Research lab, is an open-source machine learning library based on the Torch library and is used for applications such as natural language processing. Torchvision, specifically, extends PyTorch's capabilities to the realm of image processing. It provides a rich collection of pre-trained models that have been proven effective for various image-related tasks, such as object detection and classification. These models can be seamlessly integrated into existing codebases, allowing for the flexibility and ease of use that is crucial for research experimentation and prototyping (Paszke et al., 2019).

Evaluating Image Classification Models

Leveraging our understanding of precision and recall from the context of text model evaluation, we can apply these metrics to the domain of image classification with similar objectives. In image classification, precision remains a measure of the model's accuracy in predicting positive instances, while recall indicates the model's ability to capture all relevant instances within the dataset. The balance between these two metrics is often represented by the F1 score (Powers, 2011).

In Python, using libraries such as scikit-learn, we can easily compute these metrics for image classification models. The "`precision_score`," "`recall_score`," and "`f1_score`" functions from the "`metrics`" module allow us to quantify the performance of our models succinctly (Pedregosa et al., 2011). Here's a brief example of how one might calculate these metrics:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_true, y_pred)
```

To interpret the matrix, consider that the diagonal elements represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier. The higher the diagonal values relative to the off-diagonal values, the better the model's performance.

To calculate precision and recall, one can use the "`precision_score`" and "`recall_score`" functions from scikit-learn:

```
from sklearn.metrics import precision_score, recall_score
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)
```

Remember that the F1 score is the harmonic mean of precision and recall, providing a single score that balances the two metrics. It is particularly useful when seeking a balance between precision and recall, and there is an uneven class distribution (large number of actual negatives). The F1 score can be computed using the "`f1_score`" function:

```
from sklearn.metrics import f1_score
f1 = f1_score(y_true, y_pred)
```

Ethical and Practical Considerations in Image Analysis

The advent of sophisticated image-recognition models is tangible. However, these models are not without their ethical quandaries and practical limitations. A particularly poignant aspect of this discourse involves the biases inherent in image recognition systems (Buolamwini & Gebru, 2018). These biases can have far-reaching consequences, including altering how your data are labeled in a content analysis.

The X image cropping controversy serves as a case study highlighting the practical implications of such biases. In 2020, X users noticed that the platform's automatic image-cropping algorithm seemed to favor the faces of white individuals over those of black individuals (Chowdhury, 2021). This sparked widespread concern and debate over the biases embedded within the algorithm. X's cropping tool used machine learning to determine the most "interesting" part of a picture to display in a user's feed. However, due to biased training data or flawed algorithmic design, the tool exhibited a preference for lighter-skinned individuals, raising serious ethical questions about fairness and representation in automated systems.

In terms of a content analysis, we can liken this to a spurious correlation. If more images that were deemed "interesting" had white faces than black, over time, a supervised machine-learning algorithm will lock on to that feature. Because our features in image AI are lazy in a sense, meaning specifically that we extract them automatically without much supervision or intuition, these biases are especially easy to occur (Zou & Schiebinger, 2018).

Another area where biases in image recognition models have profound ethical implications is in the use of facial recognition technology for law enforcement and surveillance. Studies have revealed that some facial recognition systems have higher error rates when identifying individuals from certain racial and ethnic backgrounds (Garvie et al., 2016). This discrepancy raises concerns about racial profiling and the potential for these systems to exacerbate existing social inequalities. For instance, if a facial-recognition system is more likely to misidentify people of color, it could lead to a disproportionate number of false accusations or unwarranted scrutiny for these communities.

The consequences of such biases are not merely theoretical but have tangible impacts on individuals' lives and society at large. Biased image recognition models can perpetuate stereotypes, reinforce discriminatory practices, and erode the trust necessary for the successful deployment of these technologies. Therefore, addressing these biases is not only a technical challenge but also a moral imperative (Crawford & Calo, 2016).

Researchers and developers are exploring various strategies to mitigate bias in image recognition models. These include diversifying training datasets, implementing algorithmic audits, and developing more inclusive design practices (Gebru et al., 2018). By incorporating a broader spectrum of human diversity into the development process, the aim is to create image-recognition systems that are fairer and more representative of the global population.

Engagement with stakeholders, including those from marginalized communities, is essential in this process. Their perspectives can inform the design and implementation of image-recognition systems, ensuring that these technologies serve the needs of a diverse society. Through interdisciplinary collaboration, the integration of ethical considerations into the technical development of image recognition models can be achieved, leading to more equitable outcomes (Eubanks, 2018).

In the realm of mass-communication research, the examination of biases in image analysis is not only a technical issue but also a reflection of broader societal values and power dynamics. The case studies of X's image cropping algorithm and racial profiling in facial recognition underscore the importance of vigilant and ongoing scrutiny of these technologies. As image-recognition models continue to permeate various aspects of society, it is imperative to confront the ethical and practical challenges they pose, ensuring that these powerful tools do not become instruments of injustice (Noble, 2018).

Humans in the Loop

As we wind down this book, I hope one central theme I have put forward to you is the necessity for a human in the loop, the human coder in the content analysis. You must validate machine-learning models. It is a recognition that, despite the leaps in artificial intelligence, human oversight remains indispensable, particularly when it comes to the nuanced and context-dependent task of content analysis (Holstein et al., 2019).

Remember, codebooks are structured guides that provide clear definitions and examples for each category or theme that a model is intended to detect within images. These documents serve as the backbone of content analysis, offering a transparent and standardized framework that guides both human coders and machine learning algorithms (Neuendorf, 2017). We must evaluate our machine-learning algorithms against the codebook and treat it as if it were a human coder in a traditional content analysis.

Guidelines for Incorporating Humans in the Loop

Data Annotation

Before a model can be trained, it requires a dataset labeled with the correct annotations. As we discussed in Chapter 1, human coders are involved in this initial phase to provide these annotations, which serve as the ground truth for the model (Russakovsky et al., 2015). Annotators use the codebook as a reference to ensure consistency and accuracy in labeling. The sample size for annotation depends on the complexity of the task and the diversity of the dataset. A common practice is to start with a small but representative subset of the data, which can be gradually expanded. For instance, annotating several hundred to a few thousand images may provide a sufficient basis for initial model training, depending on the variability within the dataset.

Model Training Oversight

As the model is being trained, human reviewers should periodically check its performance against a validation set—a portion of the data reserved for testing the model's interpretations (Amershi et al., 2019). This involves reviewing the model's predictions and comparing them to the human-provided annotations. The reviewers correct any misclassifications, providing feedback that can be used to further train and refine the model. This iterative process continues until the model achieves a satisfactory level of accuracy and reliability.

Limitations of Image AI

One of the principal limitations in the current landscape of automated-image analysis is the issue of generalizability. Deep-learning models, especially those trained on large datasets, have demonstrated proficiency in recognizing and classifying images within the contexts in which they have been trained. However, the performance of these models often deteriorates when applied to new datasets or real-world scenarios that differ from the training environment (Torralba & Efros, 2011). This lack of generalizability can be attributed to overfitting, where models learn to perform exceedingly well on the training data, but fail to maintain that performance on unseen data.

Another concern that warrants attention is the interpretability of automated-image analysis models. Despite their efficacy, many deep-learning models remain opaque, often described as "black boxes" due to the difficulty in understanding how they arrive at specific outputs (Castelvecchi, 2016). This opacity is problematic, particularly in fields where understanding the decision-making process is crucial, such as in medical diagnostics or autonomous driving, but it also has implications for content analysis. The

interpretability of these models is not merely a theoretical concern but a practical one, as it affects trust and adoption by end-users who must rely on the decisions made by these systems. Hence, there is a growing demand for models that are not only accurate but also explainable, where the reasoning behind their conclusions can be comprehensively understood by humans (Ribeiro et al., 2016). For content analysis, this means we can't easily diagnose mis-classifications, or understand why our images are being labeled correctly.

The black-box nature of deep-learning models also raises ethical considerations, especially in terms of accountability and bias. The inability to fully dissect and comprehend the decision-making process of these models can lead to situations where biases present in the training data are perpetuated and amplified without the ability to easily identify or correct them (O'Neil, 2016). This can result in discriminatory practices when applied to real-world scenarios, such as facial-recognition technologies exhibiting racial or gender biases. In these ways, our models, too, may misclassify our data.

Addressing these ethical dilemmas requires a concerted effort to design models that are transparent and equitable, with mechanisms to identify and mitigate biases within the algorithms. For us as content analysts, it doubles the importance of humans in the loop. The only way we can detect these issues is with a human qualitative review.

Considering these challenges, the field of automated-image analysis stands at a pivotal juncture. As researchers and practitioners continue to push the boundaries of what is possible with deep learning, it becomes increasingly important to balance the pursuit of accuracy and efficiency with the need for generalizable, interpretable, and ethical models. The future directions of this field must include concerted efforts to develop methodologies that not only enhance the performance of these systems but also ensure their reliability and trustworthiness in the diverse applications they serve (Doshi-Velez & Kim, 2017). This will likely involve interdisciplinary collaborations, bringing together expertise from computer science, cognitive science, ethics, and domain-specific knowledge to create holistic solutions that address these multifaceted challenges.

Image-to-Text AI

I'd like to close the book with one final example of a method I think is ripe for scholars wanting to study images in a communication context. Imagine an image from a sports article depicting a female athlete in mid-stride, sprinting down the track with a determined expression. The background is a blur, emphasizing her speed. This image could be analyzed using Google's Vision AI to extract labels that describe the content (Google Cloud, 2021). Google's Vision AI might return labels such as "athlete," "running," "track and field," "competition," "female," and "motion." These

labels provide a high-level understanding of the image content, which can be used for categorizing images, searching for specific types of visual content, or analyzing trends in sports media representation.

Google's Vision AI is a powerful tool that allows developers to understand the content of an image using machine-learning models. It can detect objects, read printed and handwritten text, and even provide insight into the emotional content of the image. The cost of using Google's Vision AI depends on the number of requests made to the service. Google offers a free tier with limited usage and then charges per 1,000 requests beyond that limit (Google Cloud Pricing, 2021).

Talking to Images with OpenAI's Vision AI

Google's pre-trained models work well and are massive, but in using a tool like this, we would need to accept that the labels the model returns are predetermined by Google, meaning we can't create our own labels that perfectly match our concepts. Also, if we find that Google's Vision AI is not precise enough or has too low of recall, we cannot "fine tune" the vision AI to perfectly fit our problem, like we can with large language models (Radford et al., 2021). OpenAI's new vision API introduces a novel way of "chatting" with images. Unlike traditional image-recognition services that return predefined labels, OpenAI's Vision AI allows users to engage in a dialogue with images by asking specific questions. This conversational approach provides a more nuanced understanding of the visual content and can be tailored to the context of the image (OpenAI, 2021).

In the context of mass-communication research on gender and sport, researchers such as Cooky et al. (2013) in their study "Women Play Sport, But Not on TV: A Longitudinal Study of Televised News Media" examined the representation of female athletes in media. They were particularly interested in the portrayal of women in sports imagery, differentiating between images that depict female athletes in action, which emphasize their athletic competence, and those that portray them in passive or sexualized ways, which can undermine their athletic achievements.

Drawing from the conceptual definitions provided by Cooky et al. (2013), OpenAI's Vision AI can be prompted to classify images with more nuanced questions. For example, instead of a binary action/glamorous shot distinction, the prompts could be refined to ask, "Does this image depict the female athlete actively engaged in a sporting event?" and "Does this image portray the athlete in a manner that emphasizes non-sporting attributes such as physical attractiveness?"

By aligning the prompts with the specific research questions and conceptual definitions from the literature, OpenAI's Vision AI can provide labels that are directly relevant to the study's focus. This approach ensures that the

automated-image analysis is grounded in established academic frameworks and can contribute meaningful data to the discourse on gender representation in sports media.

GDELT: A Conceptual Overview for Image Content Analysis

The Global Database of Events, Language, and Tone (GDELT) is an expansive repository that monitors the world's broadcast, print, and web news from nearly every corner of every country in over 100 languages. It uses advanced natural language processing, machine learning, and human curation to create a structured database that captures a wide array of events, images, and narratives (Leetaru & Schrodt, 2013). Researchers can leverage GDELT to analyze trends in news coverage, including the visual portrayal of politics.

In our final notebook, to focus our analysis on images related to politics, we can filter GDELT data for articles that mention politics by GDELT's pre-baked notion of themes. It's a broad measure, but can quickly weed out irrelevant news articles. This targeted approach increases the likelihood of capturing images that are relevant to our research interest. In the notebook, I use OpenAI's Vision AI to get image descriptions that I could then parse into several codes in a codebook.[4]

Conclusion and Future Directions

As we peer into the horizon of automated-image analysis technologies, it is evident that the field is poised for significant advancements that promise to reshape the landscape of mass-communication research. The rapid evolution of unsupervised learning algorithms presents a particularly intriguing development. These algorithms have the potential to autonomously discern patterns and features in image data without the need for human-annotated training sets. This advancement could lead to a transformative reduction in the time and resources required to process and analyze large volumes of image content, thereby enabling researchers to tackle datasets of a scale previously deemed unmanageable. Remember, humans in the loop are always necessary; we must review these labels to ensure they are of quality.

These anticipated developments in automated-image analysis technologies, however, are not without their challenges. The ethical considerations surrounding the use of artificial intelligence in analyzing image content, particularly regarding privacy and bias, require careful consideration. Furthermore, the interpretive nuances that human analysts bring to the table cannot be fully replicated by algorithms, underscoring the need for a balanced approach that leverages the strengths of both human expertise and computational efficiency.

As the field of automated-image analysis matures, it is incumbent upon researchers to remain vigilant to these challenges, fostering a research environment that not only embraces technological innovation but also upholds the ethical standards and critical thinking that are the hallmarks of rigorous academic inquiry. The journey ahead is one of both excitement and responsibility, as we navigate the complex interplay between technological possibilities and the imperatives of sound, ethical research practices in the realm of mass communication. In future versions of this book, I'll be sure to focus more on images and video classification, as there are sure to be more innovations in the days to come.

With that, I close this book with an unbounded sense of optimism. While we are living in a world that is recording more media data than ever, we are now seeing for the first time that labeling and making sense of that data is more attainable than ever, even when that data is "big data."

Notes

1 OpenAI's Whisper is a deep-learning algorithm that has been trained to take audio and generate amazing transcripts. I am particularly floored by its ability to generate "any-to-english" translations. It is free, and can run inside of a basic Google Colab notebook with GPU enabled at a decent speed: https://colab.research.google.com/drive/1WLYoBvA3YNKQ0X2lC9udUOmjK7rZgAwr?usp=sharing
2 Older news-image URLs are likely to 404 due to paywalls and changes in website architecture. I would consider capturing the images for a pre-mediated time frame, like one would a streaming API.
3 As we've discussed, TensorFlow, an open-source software library for dataflow and differentiable programming across a range of tasks, is pivotal in the field of machine learning. Within TensorFlow, the Keras API emerges as a high-level neural networks library, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.
4 Remember to replace "your-api-key" with your actual OpenAI API key (https://help.openai.com/en/articles/4936850-where-do-i-find-my-openai-api-key). This script will print out the response from OpenAI's Vision AI, classifying the image according to its relevance to politics. The filtering ensures that the analysis is focused on articles related to politics, providing insights into how the media is representing it at the moment.

References

Amershi, S., Cakmak, M., Knox, W. B., & Kulesza, T. (2019). Designing AI: A human-centered approach to trustworthy AI. *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, 117–123.

Brügger, N. (2018). *The archived web: Doing history in the digital age*. MIT Press.

Buolamwini, J., & Gebru, T. (2018). Gender shades: Intersectional accuracy disparities in commercial gender classification. *Proceedings of Machine Learning Research*, *81*, 1–15.

Castelvecchi, D. (2016). Can we open the black box of AI? *Nature News*, *538*(7623), 20–23.

Chollet, F. (2018). *Deep learning with Python*. Manning.
Chowdhury, R. (2021, May 19). *Sharing our learnings about our image cropping algorithm*. X. https://blog.x.com/engineering/en_us/topics/insights/2021/sharing-learnings-about-our-image-cropping-algorithm
Cooky, C., Messner, M. A., & Hextrum, R. H. (2013). Women play sport, but not on TV: A longitudinal study of televised news media. *Communication & Sport*, *1*(3), 203–230.
Crawford, K., & Calo, R. (2016). There is a blind spot in AI research. *Nature News*, *538*(7625), 311–313.
Doshi-Velez, F., & Kim, B. (2017). *Towards a rigorous science of interpretable machine learning*. arXiv. https://doi.org/10.48550/arXiv.1702.08608
Eubanks, V. (2018). *Automating inequality: How high-tech tools profile, police, and punish the poor*. St. Martin's Press.
Garvie, C., Bedoya, A., & Frankle, J. (2016). *The perpetual line-up: Unregulated police face recognition in America*. Georgetown Law, Center on Privacy & Technology. https://www.law.georgetown.edu/privacy-technology-center/publications/the-perpetual-line-up/
Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J. W., Wallach, H., Daumé III, H., & Crawford, K. (2018). *Datasheets for datasets*. arXiv. https://doi.org/10.48550/arXiv.1803.09010
Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.
Google Cloud. (2021). *Vision AI*. https://cloud.google.com/vision
Google Cloud Pricing. (2021). *Vision API pricing*. https://cloud.google.com/vision/pricing
Grabe, M. E., & Bucy, E. P. (2009). *Image bite politics: News and the visual framing of elections*. Oxford University Press.
Highfield, T., & Leaver, T. (2016). Instagrammatics and digital methods: Studying visual social media, from selfies and GIFs to memes and emoji. *Communication Research and Practice*, *2*(1), 47–62. https://doi.org/10.1080/22041451.2016.1155332
Holstein, K., Wortman Vaughan, J., Daumé III, H., Dudik, M., & Wallach, H. (2019). Improving fairness in machine learning systems: What do industry practitioners need? *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1–16.
Kwak, H., & An, J. (2016). Understanding news geography and major determinants of global news coverage of disasters. *PLOS ONE*, *11*(5), e0155295.
LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.
Leetaru, K., & Schrodt, P. A. (2013). GDELT: Global data on events, language, and tone, 1979–2012. *International Studies Association Annual Conference*, *2*(4). http://data.gdeltproject.org/documentation/ISA.2013.GDELT.pdf
Messaris, P. (1997). *Visual persuasion: The role of images in advertising*. Sage.
Neuendorf, K. A. (2017). *The content analysis guidebook*. Sage.
Noble, S. U. (2018). *Algorithms of oppression: How search engines reinforce racism*. NYU Press.
O'Neil, C. (2016). *Weapons of math destruction: How Big Data increases inequality and threatens democracy*. Crown.
OpenAI. (2021). *OpenAI API*. https://beta.openai.com
Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., . . . & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, *721*, 8026–8037.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(2011), 2825–2830.

Powers, D. M. W. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, *2*(1), 37–63.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). *Learning transferable visual models from natural language supervision*. arXiv. https://doi.org/10.48550/arXiv.2103.00020

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144.https://doi.org/10.1145/2939672.2939778

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, *115*(3), 211–252. https://doi.org/10.1007/s11263-015-0816-y

Torralba, A., & Efros, A. A. (2011). Unbiased look at dataset bias. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 1521–1528.

Zhao, W., Chellappa, R., Phillips, P. J., & Rosenfeld, A. (2019). Face recognition: A literature survey. *ACM Computing Surveys*, *35*(4), 399–458. https://doi.org/10.1145/954339.954342

Zook, M., Barocas, S., Boyd, D., Crawford, K., Keller, E., Gangadharan, S. P., Goodman, A., Hollander, R., Koenig, B., Metcalf, J., Narayanan, A., Nelson, A., & Pasquale, F. (2017). Ten simple rules for responsible big data research. *PLOS Computational Biology*, *13*(3), e1005399. https://doi.org/10.1371/journal.pcbi.1005399

Zou, J., & Schiebinger, L. (2018). AI can be sexist and racist—it's time to make it fair. *Nature News*, *559*(7714), 324–326. https://doi.org/10.1038/d41586-018-05707-8

Appendix A

Codebook and Conceptual Definitions

Code attributes as "1" if present, "0" if absent. Follow the following definitions:

Identity-Based Negative Language. This category deals with language that either uses identity features (e.g., race, sexual orientation, gender, immigration status) in a negative manner *or* attempts to negatively situate two or more identity groups. Positive mentions of identity are not considered identity-based negative language (e.g., celebrating Gay Pride).

Inflammatory Language: This category speaks to the use of unnecessarily negative emotional language that urges audiences to be upset and/or to take action regarding a political or social issue.

Obscene Language: If a post contains any vulgar language, including any known "swear" words, no matter how mild, including "hell" or "damn," this attribute is present. Even quoted language should be considered. Identity labels used in positive ways (e.g., gay, or Muslim) are not obscene. Attempts to censor swear words (e.g., bullsh*t) are still obscenities.

Threatening Language: If a potential threat is exposed to the audience, then this attribute is present. For instance, warnings of police violence or warnings of the risks associated with illegal immigration are all examples of threats to the audience. Frequently, these threats will be suggested or implied, not directly posed to the audience.

Appendix B

Deletion Themes

OP Themes

1. Off-topic or repetitive content: not relevant to the subreddit's main theme or because they contain repetitive content that does not contribute to meaningful discussions.
2. Sensitive or controversial topics: sensitive or controversial issues are often deleted due to their potential to incite heated debates or arguments among users.
3. Privacy concerns and personal information: deleted due to the personal nature of the content and users' desire to maintain privacy or protect sensitive information.
4. Violation of subreddit rules or guidelines: do not adhere to the specific rules or guidelines of a subreddit and are often deleted to maintain a respectful and inclusive environment.
5. Promoting misinformation: promote misinformation, or conspiracy theories are often deleted to maintain the integrity of the platform and protect users from potentially dangerous information.

Moderator Themes

1. Controversial or sensitive content: controversial political figures, divisive social issues, or sensitive events are often deleted due to their potential to incite uncivil debates or violate platform policies.
2. Promoting misinformation: promote misinformation, or conspiracy theories are often deleted to maintain the integrity of the platform and protect users from potentially dangerous information.
3. Off-topic or repetitive content: not relevant to the subreddit's main theme or because they contain repetitive content that does not contribute to meaningful discussions.

4. Low-quality or repetitive content: low-quality, repetitive, or not contributing to meaningful discussions are often deleted. For example, meme-related content or humorous content and jokes.
5. Explicit content: explicit or adult content is often deleted due to its potential to violate Reddit's content policies or subreddit-specific guidelines related to user safety. For example, sexual and explicit content or discussions on intentional HIV transmission may be deleted due to their explicit nature and potential to spread misinformation.

Index