Getting to the Core of Algorithmic News Aggregators

Applying a crowdsourced audit to the trending stories section of Apple News

Jack Bandy
Northwestern University
Evanston, Illinois
johnbandy2022@u.northwestern.edu

Nicholas Diakopoulos
Northwestern University
Evanston, Illinois
nad@northwestern.edu

ABSTRACT

This paper presents a crowdsourced auditing framework for news aggregators and applies it to the trending section of Apple News. The framework audits the aggregator algorithm, determining the refresh interval and detecting the presence of "adaptation" (an aggregator presenting different headlines based on a userâĂŹs location or individual preferences). It is also used for a content audit which tabulates the distribution of news sources found in the aggregator. We deploy this framework on the trending stories section of Apple News, observing (1) a refresh interval of approximately 60 minutes, (2) adaptation at the user level, and (3) a unique distribution of news sources that prompts further investigation.

1 INTRODUCTION

Digitization, algorithmic curation, and intermediation are rapidly changing the nature of news distribution. While in the past users might subscribe directly to a newspaper or tune into a nightly brand-name broadcast, news consumption today is far more idiosyncratic, with news information delivered via homepages, search engines [6], social media, algorithmically driven aggregator pages (e.g. Google News), apps (e.g. SmartNews), chat systems and bots, smart speakers and agents (e.g. Siri, Alexa), and myriad other digital channels. Across this fragmented landscape, news organizations are trying to scrape together as much attention as they can to spread their content, build their brands, and ultimately drive some advertising revenue to sustain their businesses in what has been described as the "third wave" of journalism [1]. But the increasing power that intermediaries have over the visibility and distribution of content creates a challenging environment for publishers [10].

As one example, consider that as of 2016, 44 percent of Americans watched or read news on Facebook [7]. Recognizing this trend, many publishers focused on leveraging Facebook's News Feed to drive traffic to their websites and, in turn, generate revenue from advertisements. But in the wake of public scrutiny and criticism throughout the 2016 election, Facebook announced several revisions of their News Feed algorithm, first boosting posts from close friends [1], then deprioritizing "clickbait" links [2], and finally focusing on posts that encouraged interaction between friends and family while showing fewer "videos and other posts from publishers or businesses." [3] After Facebook's changes, overall traffic to news articles coming from the site has declined [12]. Furthermore, publishers report seeing the reach of (and advertising revenue from) their content plummet on the platform [4]. All of this has publishers looking for the next big source of traffic. One platform that's gained a fair bit of traction recently, and which is the focus of this paper, is Apple News.

Since its release in September, 2015 for all iOS 9 devices, Apple News appears to have steadily grown in popularly. In the last year publishers have reported significant referral traffic increases from the app [13]. Perhaps Apple News' most distinguishing feature in comparison to other aggregators and platforms is its reliance on humans for much of the curation. Articles are assigned to different categories that users can "like" (essentially subscribing them to those categories) in the application. In addition, the app has sections presented to all users by default, including "Top Stories" and "Trending Stories." Because the app prominently presents these sections to every user of Apple News, including a handful of stories directly on the home screen, a story's placement in these sections presumably generates significant attention. And while the editorial staff deliberately chooses which "Top Stories" are shown on a day-to-day basis, "Trending Stories" are still algorithmically selected [13]. This study specifically focuses on the algorithmically selected stories in the trending section of the app. We should note, however, that it remains unclear to what extent there may still be human oversight of Apple News' trending section.

While previous research has considered issues of coverage and ranking bias on platforms such as Google News [8, 11], Google search [6], Twitter [4], and even of the human-curated portions of Apple News [3] no work has yet been done to audit the algorithmic portions of Apple News, namely, the trending stories. Given that Apple News already drives significant attention to news information, it is important to know *where* the platform is driving users' attention, and whether it is concentrating (or fairly distributing) attention across the set of news sources it curates. The goal of this paper is to formally audit the "Trending Stories" section of the Apple News app in order to better understand it as a mediator of attention. In particular, the study presented here contributes (1) a characterization of the news sources and their relative prevalence on Apple News and (2) a crowdsourcing framework the can be applied to auditing similar news aggregation apps in the future. In the next sections we outline our methods and results for our audit of the Apple News trending algorithm and the content it surfaces.

2 AUDITING METHODS

There were three primary research questions driving our audit. In terms of the curation algorithm itself, we wanted to know (1) how

[1] https://newsroom.fb.com/news/2016/06/news-feed-fyi-helping-make-sure-you-dont-miss-stories-from-friends/
[2] https://newsroom.fb.com/news/2017/06/news-feed-fyi-showing-more-informative-links-in-news-feed/
[3] https://newsroom.fb.com/news/2018/01/news-feed-fyi-bringing-people-closer-together/
[4] https://slate.com/technology/2018/06/facebooks-retreat-from-the-news-has-painful-for-publishers-including-slate.html

often does the algorithm refresh stories?, and (2) does the algorithm adapt stories to users based on location or individual preference? The answers to both of these questions were then consequential to how we collected data to address the third question related to the actual content surfaced by the curation algorithm: (3) what is the distribution and concentration of news sources appearing in the trending section? We next describe our general approach to gathering data for the audit, followed by the specifics of the data collection methods used to address each research question.

2.1 Data Collection Method

The most straightforward auditing method for many news services and applications is a "scraping audit," [14], in which an application programming interface (API) allows direct access to content via API calls. Since Apple News has no public API, we initially attempted to find its hidden API using a proxy server to inspect packets traveling in and out of the iOS App as well as the macOS version of Apple News. Our investigation indicated that Apple News exchanges data with Apple's servers using SSL pinning, a security technique that blocks all API calls unless an application-specific security certificate is used. This prevented us from scraping news stories with an API call or using any kind of computer program to collect data.

With automated data collection techniques effectively blocked, a scraping audit was not possible. We instead turned to the crowd as our primary means of data collection for the audit, described in [14] as a "collaborative audit" or simply a "crowdsourced audit." In a crowdsourced audit, human testers access the data and then provide their respective data points to a central repository for aggregation and evaluation. Such techniques have been effectively used in journalism to, for instance, study personalization of Google Search results and investigate Facebook's friend recommendation algorithm [5]. Our study required the collection of headlines from the Trending Stories section of the Apple News app, so we asked human testers to take a screenshot showing those headlines. We then used these screenshots to investigate several questions about the algorithm behind Apple News' Trending Stories. In our subsequent analyses we verified screenshots manually to ensure they were properly taken, which we define as (1) showing the trending section fully, (2) showing all four stories, and (3) being taken at the correct time.

For our crowdsourcing we leveraged the Amazon Mechanical Turk (AMT) platform for which we designed Human Intelligence Tasks (HITs) that allowed us to collect the data necessary for our audit. A crucial step in using AMT ethically is paying users fair wages [9]. We aimed to compensate workers' time according to the United States federal minimum wage of $7.25 per hour [5]. To do this, we deployed a pilot version of our HIT to fifteen users and measured the median time to completion, which was 6 minutes. We round up and consider $0.75 to be the *minimum* pay for the work of taking a single screenshot of the app for our data collection. In cases where we wanted to increase the chances of a task being completed, we used a "high-reward" rate of $2.00 for a screenshot.

Some of our HITs (those designed to audit the algorithm for personalization and localization) require multiple users to take synchronized screenshots, so that stories appearing at the same time can be compared across other variables such as location. Synchronization in crowdwork is challenging since crowdworkers perform tasks on their own schedule. We follow an approach from [2], in which we essentially pay workers to wait until a designated time to perform the task. For example, in some cases, users checking in at 10:36am, 10:42am, and 10:51am would all be instructed to wait until 11:00am to take the screenshot. We also required workers to wait in some of our multi-screenshot tasks, with wait times ranging from 5 minutes to 45 minutes. When paying workers to wait, we calculated pay by the base single-screenshot rate ($0.75) plus minimum wage for the amount of waiting time. For example, if the user was taking three screenshots that were spaced 30 minutes apart, we would calculate their compensation as follows: `compensation = base_pay + (wait_time * wage)`. For example, a worker waiting for 30 minutes between each of two screenshots uploaded (a total of one hour of waiting) would be compensated `$0.75 + (1hr * $7.25) = $8.00`.

2.2 Auditing the Algorithm

2.2.1 Determining the refresh interval. In order to efficiently and comprehensively audit other aspects of the Trending Stories algorithm, we first needed to determine its update frequency using screenshots from crowdworkers. Intuitively, one could refresh the app and take a screenshot every minute, noting when changes occurred. But by determining the refresh interval this allows us to be more efficient (yet still comprehensive) in subsequent data collection.

One complication is the possibility that a refresh might not yield different stories every time, not because the app hasn't technically refreshed, but because the stories that are trending simply haven't changed. For example, if the app refreshes trending stories at 3:30am, but the trending stories are the same at 3:06am and 3:36am, we would see no evidence of the refresh. Therefore, it is best to think of the question as "what is the shortest amount of time in which the algorithm *can* change trending stories," rather than "how often does the algorithm change its trending stories."

Here we leverage the Nyquist-Shannon sampling theorem, which indicates that in order to fully represent a signal digitally, we must sample at frequency `2B` if the frequency of change is `B`. In other words, if the app updates stories every `N` minutes, we must sample every `(1/2)N` minutes to guarantee that we observe all changes in stories. With this in mind, our approach was to collect three screenshots in a row from individual users. If the screenshot interval was every N minutes, three screenshots with all different headlines would show that the app can update stories every `(2N - 1)` minutes. We found this approach to be slightly counterintuitive, so we describe the insufficiency of alternative approaches below.

Insufficient approach 1: collect single screenshots from *multiple users*, then analyze all screenshots together to see when stories change. If Alice's screenshot at 12:04 PDT shows different stories than Bob's screenshot at 12:16 PDT, it is possible this difference is owed to personalized or localized news stories, rather than a refresh of the trending stories. Without first knowing whether the app localizes or personalizes stories, we cannot combine screenshots from multiple users in our analysis.

[5] https://www.dol.gov/general/topic/wages/minimumwage

Insufficient approach 2: collect and analyze *two* screenshots from individual users, and see if the stories change from the first screenshot to the second. This data could show specific times at which the app refreshes stories, but not the interval at which those refreshes occur. For example, if the app refreshes at 12:00 PDT, we could observe the change from numerous sampling intervals (11:59 PDT and 12:01 PDT; 11:30 PDT and 12:30 PDT; etc.). And, once again, due to the possibility of adaptation, we could not combine screenshot pairs from multiple users to make further inferences.

Because of the issues associated with combining screenshots from multiple users and observing only two screenshots from a user, we developed an approach for determining the refresh interval as follows.

Algorithm 1 Determine if the trending section refreshes at a given interval. We ran this algorithm several times, decreasing *interval_to_test* by 10 until the algorithm returned True.

Require: *interval_to_test*, an integer representing minutes
  sampling_interval ← (*interval_to_test* / 2)
  {collect three screenshots (x, y, z) exactly *sampling_interval* minutes apart from each user in N}
  for all users in N with screenshots (x, y, z) **do**
    if (x and y and z all display different headlines) **then**
      return False {The stories refresh more frequently than *interval_to_test*}
    end if
  end for
  return True {No evidence shows stories can refresh more frequently than *interval_to_test*}

To help understand this approach, imagine that the trending stories refresh every 30 minutes (at 1:00, 1:30, 2:00, and 2:30), and we hypothesize that Apple News can refresh every 40 minutes. Thus, we ask users to take three screenshots, twenty minutes apart. Some screenshot sets would contain two screenshots with the same stories and one screenshot with a change (for example, screenshots taken at 1:05, 1:25, and 1:45). However, with adequate sampling, one or more screenshot sets would contain three screenshots with different stories (for example, 1:25, 1:45, and 2:05), or a "double change." At the point where we observe two changes across three screenshots we falsify our initial hypothesis and update it to hypothesize that the refresh interval is shorter, perhaps 30 minutes. We then repeat the experiment with the new hypothesis. Once the screenshot interval is *half* the refresh interval (15 minutes in this example), it is impossible to take three screenshots showing three different sets of stories.

Crucially, while our hypothesis of a given interval can stand after this algorithm runs, it is not guaranteed. In other words, we could still encounter evidence to falsify our current hypothesis, thus our findings hold a degree of uncertainty determined by N, the number of users providing screenshot sets.

2.2.2 Determining Adaptation. Apple News users can "like" specific topics and publishers to populate the app with personalized stories. We wanted to determine if any personalization and/or localization applied to the "Trending Stories" section, or if every user sees the same list of trending stories. The method works as described in Algorithm 2.

Algorithm 2 Algorithm to check for adaptation

Require: set of screenshots S from crowdworkers
Ensure: all screenshots in S were taken at the same time
  for all possible (x, y) pairs in S **do**
    if x and y display different headlines **then**
      return True {Adaptation observed}
    end if
  end for
  return False {No adaptation observed}

If Algorithm 2 returned true, we would then run another algorithm to determine whether location accounted for the adaptation, or whether other factors such as individual preferences were at play. To remove both time and location from possible causes for different headlines, we would run Algorithm 2, but add the stipulation that all screenshots in S were taken in the same geographic area (e.g. zip code).

2.3 Auditing the Content

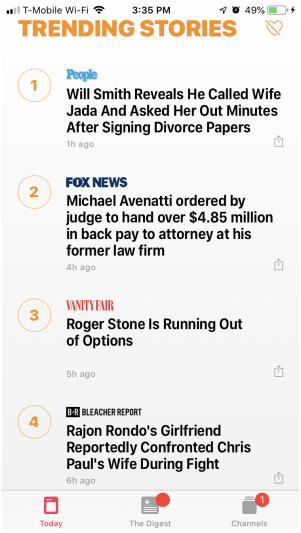

Figure 1: An example of a screenshot taken by an AMT worker

2.3.1 Screenshot Processing. AMT workers uploaded screenshots to imgur, where we were able to access them directly via the site's API. Once downloaded, we fed screenshots through Google's tesseract OCR engine [15] to extract all text contained in the screenshot. With some basic parsing, we were able to determine the four headlines displayed.

While tesseract handled the headlines fairly well, it was unable to consistently transcribe the names of news sources, which are displayed as logos in the app (see figure 1). These logos present a more difficult OCR task than the headlines because, unlike the four headlines, they vary in size, font, shadowing, and character overlap.

Since we could not use tesseract to determine the source of each story, we modified our approach to find the URL for the story and use that to determine the source. For each story, a python script scraped Google search results [6] and NewsAPI [7] with the exact headline as the query. If the top result from Google pointed to the same URL as the top result from NewsAPI, that URL was linked to

[6] Using a library available at https://github.com/abenassi/Google-Search-API
[7] https://newsapi.org/

the headline in our database; otherwise, the headline was flagged for inspection, and we manually ensured the correct URL was linked to the headline.

3 AUDIT RESULTS

In the following subsections we describe the specific data collected to audit the algorithm and content of Apple News.

3.1 Algorithm Results

3.1.1 Refresh Interval. We ran the algorithm described in 2.2.1, starting with 90 minutes as our hypothesis for the update interval (and thus 45 minutes as the screenshot interval) and collecting screenshots from 5 users at each hypothesis. At a 70-minute hypothesis, none of the 5 users uploaded screenshots showing double changes. To be sure we assigned the task to 15 additional users and we did observe double changes. We decreased the hypothesis to 60 minutes and deployed the task to 20 more users, from which none of the screenshot sets showed a double change. We therefore suggest that the Trending Stories algorithm refreshes stories up to once per hour.

3.1.2 Adaptation. We ran three tests for adaptation, hypothesizing that Trending Stories does not localize or personalize stories within the United States.

In the first test, we ran the algorithm described in 2.2.2, making 15 screenshot tasks available to workers for the target screenshot time. Because we wanted as many samples as possible, we used our high-reward rate and paid $2.00 for a single screenshot (almost 3 times the base pay). Of the tasks deployed, 10 workers correctly took a screenshot of trending stories at 1:00pm PDT on October 19th. The 10 screenshots all showed the same headlines, so the hypothesis stood.

We also deployed two tests on the morning of October 31st, one with a target screenshot time of 6:05 AM PDT, and one with a target time of 7:35 AM PDT. These times were specifically chosen to give the Apple News app and servers time to refresh, update, and push out new trending stories in case this process was occurring at the top of each hour. Again, we used our high-reward rate to entice workers and collect as much data as possible. In the first test (6:05 AM PDT), we collected 13 properly-taken screenshots. 12 screenshots contained the same four stories in the same order, while 1 screenshot contained two of those stories *and* two additional unique stories. In the second test (7:35 AM PDT), we collected 10 properly-taken screenshots. This time, only 7 screenshots contained the same headlines in the same order. We observed 3 different combinations of headlines displayed at 7:35 AM PDT. Furthermore, 2 workers from the same zip code uploaded screenshots showing different headlines, suggesting that adaptation occurs at the *personalization* level, not at the localization level.

Our interest is mainly in whether the algorithm personalizes trending stories. However, the observed personalization could also result from user customization (e.g. a user blocked a source like CNN, and therefore no CNN stories would appear in their trending section). There are other possible explanations for our observations, as there are a number of ways our data may have been insufficient or inaccurate:

(1) We ask users to take screenshots at minute-specific times such as 5:00, not second-specific times 5:00:00. Since our method puts screenshots from 5:00:00pm and 5:00:48pm in the same bin, screenshots we consider to be taken "at the same time" often have been taken at slightly different times separated by up to 60 seconds, and it's possible the algorithm may have refreshed stories in the meantime.
(2) Users were asked to reload the news app immediately before taking the screenshot. If a user failed to reload the app, cached stories from a previous time may show up in the trending section instead of the most up-to-date stories. This might have caused screenshots from different users taken at the same time to show different stories.

3.2 Content Results

3.2.1 Extended Data Collection. To characterize the news sources represented in the trending section, we collected data over a 72-hour period. Starting at 8am CDT on October 22nd, we published HITs for Apple News screenshots. Based on our findings in 3.1.1, ideally we would collect one screenshot every hour for a total of 72 screenshots spaced apart by exactly 60 minutes each. However, if we were to publish all 72 HITs at once, we would not see an even distribution of tasks completed over time, rather, we would likely see most HITs completed within the first day, and sparse data for the last two days.

To spread out samples, we released one high-reward screenshot task every half hour for the full 72-hour period (recall from section 2.1 that we consider $2.00 our "high-reward" rate for a single screenshot). As mentioned previously, when HITs are released, most are completed early in an initial cluster. With our periodic-release approach, publishing one task every half hour, we mitigate the early completion effect. Also, the high reward incentivizes workers to accept the task soon after seeing it, which helps us avoid skewed data in the other direction (i.e. many workers completing the task near the expiration time).

To further help spread out data collection, we asked users to wait until the next even half-hour to take the screenshot. Unlike in section 3.1.1, where we paid users for their waiting time in between multiple screenshots, this task did not pay users to wait. Because AMT workers generally work on multiple tasks simultaneously [9], we link to an alarm that sounds one minute before the desired screenshot time, allowing workers to comfortably complete other HITs until the alarm sounded and it was time to take the screenshot.

With this HIT release system in place, we collected 156 total assignments of our screenshot task over a 72-hour period. This included one assignment deployed every half-hour, and an initial batch of assignments.[8] We eliminated 40 assignments that were completed incorrectly (e.g. no link to the screenshot, screenshot of incorrect section) and ended up with 116 screenshots. There were a total of 23 hours in which we received no screenshots, however, while there were 31 hours in which we receive more than 1 screenshot, a limitation we address in the future work section. For intervals in which we had multiple screenshots we randomly sampled one that we used for our content analysis (though this

[8]The Amazon BOTO API requires that HITs have a certain number of initial assignments in order to add assignments later on.

may introduce some noise due to the small amount of adaptation we observed).

Based on the data collected we observed 25 unique news sources and 103 unique articles in total. The unique sources included: ABC News, Bazaar, Bleacher Report, BuzzFeed (not BuzzFeed News), CNN, Fox News, Hollywood Reporter, Huffington Post, Mashable, National Geographic, Newsweek, NPR, NY Mag, NY Times, People, Politico, Rolling Stone, The State, Time, Vanity Fair, Vogue, Vox, Washington Post, Womens Health Magazine, and WSJ. Many of these focus on soft news. The top news sources in terms of proportion of unique stories per source are shown in Figure 2. Fox News tops the list, however, celebrity and entertainment sources (e.g. People, Vanity Fair) as well as sports (e.g. Bleacher Report) are also highly prevalent. The top 3 sources represent more than half of all unique stories observed indicating a high degree of concentration.

The average time a story spends in the trending section is 2.1 hours (M=2; SD=1.2, Max=5), indicating that there is fairly rapid turn-over of trending stories but that some stories may persist for several hours. In one case, Bleacher Report, there were four unique stories but each was only observed once.

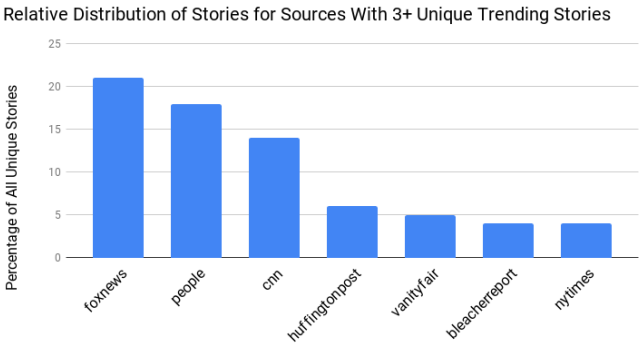

Figure 2: Distribution of unique stories over sources.

4 SUMMARY AND FUTURE WORK

We designed a crowdsourced framework for auditing the algorithm behind Apple News' trending section, and deployed the framework to answer questions related to the algorithm (refresh interval and user adaptation) as well as the algorithm's output (source distribution). Our audit showed that Apple News updates stories up to once per hour. Results also indicate that Apple News may incorporate some form of content adaptation which is most likely personalization and not localization, however additional work must be undertaken to confirm whether these results are a function of user-driven customization (i.e. blocking sources).

Our extended data collection and analysis of content suggested a skewed source distribution, with just three sources accounting for more than half of unique stories observed (top one being Fox News). The results also indicate that Apple News Trending skews towards soft news such as celebrity, fashion, entertainment, and sports, however it does also include some news from Fox News, CNN, NY Times, NPR, and the Washington Post. Lastly, a given story appears to stay "trending" for about two hours, with some small variation based on source. These findings are somewhat limited by our data, namely in that (1) data only represents a 72-hour period and (2) there are several spans from that period in which we have no screenshots and therefore no data. This underscores a limitation of the crowdsourcing approach: we cannot guarantee data collection during every interval. However, we hope to adapt our HIT release schedule in the future to better manage the flow of data and improve coverage.

Even with limitations in mind, our findings prompt further research questions.

- *Why are some news sources significantly more prominent in the trending section?* For example, this could be due to some publishers releasing more stories to Apple News, users clicking on stories from a given source more often, overall popularity on the web, or a combination of these factors.
- *Is there a similarly skewed distribution of "impressions" over news sources?* In additional to clarifying the distribution of unique stories in the trending section, we want to measure source visibility with other metrics.
- *Is there human oversight involved in the trending section?* Apple News' human curators are known to curate the Top Stories in the app [13], but it is unclear whether there is a human screening phase for the trending section.

We plan to collect data over a longer time period to further investigate these questions. Finally, we plan to apply and generalize our crowdsourced audit approach to other news aggregators.

REFERENCES

[1] Emily J Bell, Taylor Owen, Peter D Brown, Codi Hauka, and Nushin Rashidian. 2017. The platform press: How Silicon Valley reengineered journalism. (2017).
[2] David R & Miller Robert C Bernstein, Michael S & Karger and Joel Brandt. 2012. Analytic methods for optimizing realtime crowdsourcing. *arXiv:1204.2995* (2012).
[3] Pete Brown. 2018. Study: Apple News' human editors prefer a few major newsrooms. https://www.cjr.org/tow_center/.
[4] Abhijnan Chakraborty, Saptarshi Ghosh, Niloy Ganguly, and Krishna P Gummadi. 2015. Can trending news stories create coverage bias? on the impact of high content churn in online news media. In *Computation and Journalism Symposium*.
[5] Nicholas Diakopoulos. 2018. The Algorithms Beat. Unpublished Manuscript. Available at http://www.nickdiakopoulos.com/wp-content/uploads/2018/04/Diakopoulos-The-Algorithms-Beat-DDJ-Handbook-Preprint.pdf.
[6] Nicholas Diakopoulos, Daniel Trielli, Jennifer Stark, and Sean Mussenden. 2018. I Vote For - How Search Informs Our Choice of Candidate. *Digital Dominance: The Power of Google, Amazon, Facebook, and Apple* (2018).
[7] Jeffrey Gottfried and Elisa Shearer. 2016. *News Use Across Social Medial Platforms 2016.* Pew Research Center.
[8] Mario Haim, Andreas Graefe, and Hans-Bernd Brosius. 2018. Burst of the Filter Bubble? *Digital Journalism* 3 (2018), 330–343.
[9] Kotaro Hara, Abigail Adams, Kristy Milland, Saiph Savage, Chris Callison-Burch, and Jeffrey P Bigham. 2018. A Data-Driven Analysis of Workers' Earnings on Amazon Mechanical Turk. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 449.
[10] Rasmus Kleis Nielsen and Sarah Anne Ganter. 2018. Dealing with digital intermediaries: A case study of the relations between publishers and platforms. *New Media & Society* 20, 4 (2018), 1600–1617.
[11] Seth C Nechushtai, Efrat & Lewis. 2018. What kind of news gatekeepers do we want machines to be? Filter bubbles, fragmentation, and the normative dimensions of algorithmic recommendations. *Computers in Human Behavior* (2018).
[12] Nic Newman, Richard Fletcher, Antonis Kalogeropoulos, David AL Levy, and Rasmus Kleis Nielsen. 2017. Reuters Institute Digital News Report 2018. (2017).
[13] Jack Nicas. 2018. Apple News' Radical Approach: Humans Over Machines. https://nyti.ms/2JeWfTJ.
[14] Christian Sandvig, Kevin Hamilton, Karrie Karahalios, and Cedric Langbort. 2014. Auditing algorithms: Research methods for detecting discrimination on internet platforms. *Data and discrimination: converting critical concerns into productive inquiry* (2014), 1–23.
[15] Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, Vol. 2. IEEE.