Detecting Hashtag Hijacking for Hashtag Activism

Pooneh Mousavi
University of Texas at Dallas
pxm153230@utdallas.edu

Jessica Ouyang
University of Texas at Dallas
jessica.ouyang@utdallas.edu

Abstract

Social media has changed the way we engage in social activities. On Twitter, users can participate in social movements using hashtags such as #MeToo; this is known as hashtag activism. However, while these hashtags can help reshape social norms, they can also be used maliciously by spammers or troll communities for other purposes, such as signal boosting unrelated content, making a dent in a movement, or sharing hate speech. We present a Tweet-level hashtag hijacking detection framework focusing on hashtag activism. Our weakly-supervised framework uses bootstrapping to update itself as new Tweets are posted. Our experiments show that the system adapts to new topics in a social movement, as well as new hijacking strategies, maintaining strong performance over time.

1 Introduction

Social media has changed the way we live, trade, share news, and engage in social activities. Twitter is one of the most popular social networks, where users post short textual messages called "Tweets." A hashtag (#) before a particular keyword or phrase in a Tweet is used to categorize the Tweet, helping users find topics that are of interest to them.

One of the achievements of social media is reshaping and re-scaling engagement in social movements via *hashtag activism*. Yang (2016) defines hashtag activism as large numbers of social media posts using a common hashtagged phrase with a social or political claim. Some popular hashtag activism movements include "#MeToo," a movement against sexual harassment and assault, and "#BlackLivesMatter," which campaigns against violence and systemic racism towards African Americans. These hashtags help engage people in social movements by raising awareness on a larger scale and by giving opportunities for those with access limitations, like the physically challenged, to participate.

Unfortunately, hashtag activism is also a good target for spammers. *Hashtag hijacking* occurs when users "[use] a trending hashtag to promote topics that are substantially different from its recent context" (VanDam and Tan, 2016) or "to promote one's own social media agenda" (Darius and Stephany, 2019). While the detection of spam Tweets in general is an important issue, the detection of spam related to social movements is of even greater importance because it targets excluded or marginalized groups.

We present a weakly-supervised, bootstrapping framework to detect Tweet-level hashtag hijacking targeting specific social movements, using a combination of features based on the Tweet text, use of other hashtags, replies, and user profile. Our experiments focus on #MeToo, but our methodology can be applied to any hashtag. Prior work on hashtag hijacking has focused on general trending hashtags like #job or #android and could not adapt over time to attacker strategies; these approaches were unable to account for changes in hashtag use over time. Ours is the first self-updating approach to be developed for detecting hashtag hijacking at the Tweet level. Our main contributions are as follows:

- A new dataset of #MeToo Tweets from October 2017 through May 2020[1].
- A bootstrapping framework to detect hashtag hijacking that can adapt over time to hijackers' changing strategies.

2 Related Work

Hashtag hijacking is a relatively new problem, and there is little prior work on the task.

Previous studies have analyzed cases of political hashtag activism spamming ("hacktivism"), which

[1] https://github.com/poonehmousavi/Detecting-Hashtag-Hijacking-for-Hashtag-Activism

Proceedings of the 1st Workshop on NLP for Positive Impact, pages 82–92
Bangkok, Thailand (online), August 5, 2021.

often involves cyberbullying (Taylor, 2005; Hampson, 2012; Deseriis, 2017; Solomon, 2017), emphasizing the destructive role of spamming on political movements and protests and how it can change the direction and goals of the targeted movement. Lindgren (2019) identifies noise and trolling as the main challenges facing hashtag activism movements, and Kalbitzer et al. (2014) discusses on the harmful effects of excessive unwanted information on social media, which can even affect the physical condition of vulnerable users. Bode et al. (2015) evaluate the composition of political networks on Twitter; they find that hashtag "hashjacking (encroaching on opposition's keywords to inject contrary perspectives into a discourse stream)" to be one of the main types of strategic political communication on Twitter.

Few studies have investigated computation pipelines to detect hashtag hijacking. Prior work focuses exclusively on general trending hashtags and cannot adapt over time to attacker strategies.

Jain, Agarwal, and Pruthi (2015) proposed an unsupervised framework for detecting hashtag hijacking. They argued that hijacked Tweets use different words than do the more common relevant Tweets. Jain et al. grouped trending hashtags into general categories, such as *technology*, *entertainment*, and *politics*, and calculated words' TF-IDF scores at the category level. They then predicted whether or not a Tweet using a given hashtag was hijacked based on its word overlap with its category's word list. By using categories, rather than individual hashtags, Jain et al. were able to increase the amount of data for calculating their scores, and also to determine which categories of hashtags were more likely to be hijacked. In contrast, our goal is to focus on a specific hashtag associated with social activism.

Van Dam and Tan (2016) applied topic learning and time series analysis to the hijacking task for trending hashtags. They analyzed each hashtag's distribution of topics over time: if a hashtag's topic distribution in a one-day window differed significantly from its previous distribution, the hashtag was considered hijacked. Van Dam and Tan's approach operates at the level of hashtags and does not attempt to predict whether or not a specific Tweet is hijacked. Like Jain et al., they assume that a hashtag's topic distribution is constant over time, and that changes in topic indicate hijacking; in contrast, our work assumes that the topics associated with a social activism hashtag can shift over time.

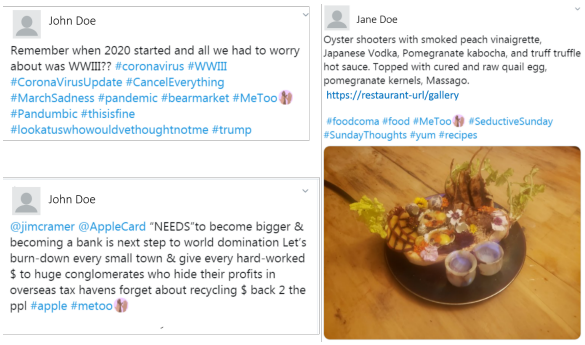


Figure 1: Hijacked #MeToo Tweets. Usernames are masked to protect the privacy of the users.

Virnami et al. (2017) trained a fully supervised neural network to detect hashtag hijacking. They extracted Tweet-level features, such as the relatedness among hashtags used and information about the user account that posted the Tweet, and trained a feed-forward network to classify Tweets as hijacked or not. Like Jain et al. and Van Dam and Tan, Virnami et al. focused on trending hashtags; they used a manually-labeled dataset of ten thousand Tweets corresponding to the top ten trending hashtags. Like Jain et al., Virnami et al. treat hijacking as a general problem unrelated to any specific hashtag; their features are independent of the hashtags used in a Tweet, allowing them to train a single neural model on a much larger dataset than would be available for any individual hashtag.

Twitter spam detection is a related area of work. Rather than detecting unrelated Tweets that hijack a given hashtag, spam detection is a more general task: determine whether or not a Tweet is spam, regardless of the hashtags it uses. Most existing techniques for spam detection can be categorized into approaches that focus on user-level features to identify spammers (Wang, 2010; Yardi et al., 2010; Lee et al., 2010), those focus on Tweet-level features to identify spam Tweets ((Gao et al., 2012)), and hybrid approaches that use a combination of features based on both Tweet and user (Sedhai and Sun, 2018; Hu et al., 2014, 2013).

A related line of research is the relationship between spam Tweets and the hashtags they use; Sedhai and Sun (2017) analyzed hashtags in spam Tweets based on their frequency, position, orthography, and co-occurrence counts. It is important to emphasize the difference between *hijacked* and *spam* Tweets. Tweets are hijacked in terms of a specific hashtag; not all hijacked Tweets are spam. Figure 1 shows examples of hijacked #MeToo Tweets

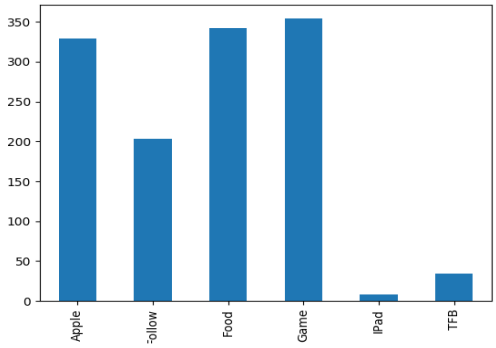


Figure 2: Tweet distribution over spam categories.

that are not spam in the general sense.

3 Data

We use data from #MeToo, a movement used by women to share their experiences with sexual harassment. The online #MeToo movement started in October 2017 with actress Alyssa Milano's Tweet about sexual abuse allegations against Harvey Weinstein. #MeToo has since become widespread and a target for hijackers to increase their own visibility and promote their products.

3.1 Data Collection

There is an enormous and continuously growing number of #MeToo Tweets, most of which are not hijacked. Thus, the first challenge addressed in our work is to collect enough hijacked Tweets for our seed set, before the Tweets are even labeled. We use the Twitter spam analysis of Sedhai and Sun (2017) to create a list of hashtags that are likely to occur in spam Tweets (Table 1). Using the Twitter API, we find Tweets containing both #Metoo and at least one of these spammy hashtags. After removing duplicates (Tweets containing multiple spammy hashtags), we are left with 1370 Tweets that are likely to be hijacked. Figure 2 shows the distribution of these Tweets over spam categories.

Note that, while a Tweet may hijack a hashtag without being spam — hijacking occurs when the hashtag is used to boost visibility for any unrelated topic, not just spam topics — a spam Tweet that uses the #MeToo hashtag is almost certainly hijacked. Thus, our collected spammy Tweets are likely to be cases of #MeToo hijacking, although they are not necessarily a representative sample of all hijacked #MeToo Tweets. From the 1370 collected Tweets, we randomly sample 100 for test set and use the rest for training and validation.

For non-hijacked Tweets, we collect 500 #MeToo Tweets from each month between October 2017, when the online #MeToo movement started, and November 2019. We remove retweets and replies, for a total of 12,892 Tweets. Since non-hijacked Tweets are much more common than hijacked Tweets, we expect most of this collection to be genuine #MeToo Tweets that capture the hashtag's use over the course of its lifespan. We randomly sample 1500 and 100 of these Tweets, evenly distributed over the 25 months, merge them with the potentially hijacked Tweets described above, and remove any duplicates, for a seed set of 2770 #MeToo Tweets for training and validation and 200 Tweets for testing; we expect roughly half of these seed Tweets to be hijacked.

3.2 Data Annotation

We use crowdsourcing on Amazon Mechanical Turk (AMT) to label our collected Tweets. #metoo was a popular tag before the movement and had a different meaning, but since the #MeToo movement has had such a large impact on popular culture, we assume in this work that anyone using the #MeToo hashtag after October 2017 would be aware of its new meaning. We consider anything related to the #MeToo, including criticism, to be non-hijacked; we ask workers to label Tweets as "related" (not hijacked) if they are relevant to the #MeToo movement, "unrelated" (hijacked) if they are irrelevant to #MeToo, or "hard to tell" if it is difficult to decide; details of the task are in Appendix A.

For each Tweet, we obtain labels from 7 AMT workers and take the majority vote among them; we break ties by randomly selecting one worker as the tie breaker. Ties happen when there are equal numbers of 'valid' and 'hijacked' votes, eg. 3 'hijacked,' 3 'valid,' and 1 'hard to tell.' Table 2 shows statistics for the distribution of data over the three labels, hijacked, non-hijacked (which we henceforth call *valid* for clarity), and hard to tell, as well as the inter-annotator agreement on the task.

3.3 Noisy Labels

On examining the AMT statistics in Table 2, we find that interannotator agreement is low: Fleiss's κ of 0.212 and 0.168 on the training and test sets, respectively. One possible cause for low agreement is that our AMT task asks workers to read and judge a single Tweet. Because the task takes very little time to do, workers may be tempted to answer randomly, without putting much effort into the task.

Category	Hashtags
TFB	#TFB, #TeamFollowBack, #FollowGain
Food	#food, #foodporn
Follow	#follow4follow, #followforfollow, #likeforlike, #like4like
Apple	#apple, #iphone
IPad	#ipad, #ipadgames
Game	#PS4live, #Gamer, #Gaming, #games, #GameNight, #VideoGames

Table 1: Spammy hashtags for collecting hijacked Tweets.

Dataset	Total	Valid	Hijacked	Hard to Tell	Agreement
AMT Training	2770	1867	830	73	0.212
Snorkel Training	2770	1603	1158	9	-
AMT Test	200	144	51	5	0.168
Expert Test	200	104	85	11	0.389
Expert Validation	200	117	74	9	0.450
Expert Live Samples	380	212	149	19	0.340

Table 2: Data annotation statistics. AMT Training and Test are produced by seven workers on Amazon Mechanical Turk. Snorkel Training is the final training data that we use to train our model; Expert Test, Validation, and Live Samples are produced using two expert annotators (Section 3.3). The Agreement column shows Fleiss's κ for AMT and Cohen's κ for Expert.

To address the issue of low agreement and questionable annotator trustworthiness, we turn to expert annotations. We train six expert annotators, graduate students from our university's computer science department. For the relatively small test and validation sets, we relabel the entire set using these expert annotators. We assign two annotators to each Tweet; for ties, we ask a third annotator to label the Tweet and break the tie. As shown in Table 2, the expert annotators achieve higher inter-annotator agreement, and the relabeled test set is more balanced between *hijacked* and *valid*.

As the training set is much larger, using expert annotators to label the whole dataset would be time-consuming and expensive. Therefore, we need an approach to reduce the noise or lessen its effect on the existing AMT-labeled data, rather than re-annotating it entirely. We use Snorkel (Ratner et al., 2017), a framework for building and managing training datasets. The Snorkel framework takes user-defined labeling functions, learns weights for each of these functions, and generates the final label using a weighted vote among the functions. We use three different types of labeling functions: a keyword-based function, a model-based function, and crowdworker-based functions.

- **Keyword-based.** Labels Tweets containing any hashtag from Table 1 as *hijacked*.
- **Model-based.** We use feature-based submodular optimization Wei et al. (2014) to choose a subset of 200 Tweets that we annotate using the expert annotators and use to train a logistic regression model. The general form of a feature-based submodular function is

$$f(X) = \sum_{d=1}^{D} \phi \left(\sum_{i=1}^{N} X_{i,d} \right)$$

where f is an objective function that uses the concave submodular function ϕ and is operating on a data subset X that has N examples and D feature dimensions. Maximizing f encourages diversity and coverage of the features within the chosen subset.

We assign one feature for each AMT worker and use the Apricot submodular data selection framework (Schreiber et al., 2019) to solve. Tweet selection is greedy: in each iteration, we choose the Tweet with the most gain. After selecting 200 Tweets, we achieve full coverage of the entire feature space, and these 200 Tweets form the validation set, which we label with the expert annotators and use to train a logistic regression model. The features are the worker IDs of the AMT workers, and the model learns weights for each worker based on how well they agree with the experts.

- **Crowdworker-based.** We consider each AMT worker as a single labeling function that returns the label submitted by that worker for a given Tweet, or abstains if that worker did not submit a label for that Tweet.

From Table 2, we see that the Snorkel method increases the number of *hijacked* Tweets in the training set, balancing it similarly to how the expert annotations balanced the test set.

4 Methodology

To detect hashtag hijacking, we present a weakly-supervised, continuously updating approach inspired by the work of Sedhai and Sun (2018) for detecting Twitter spam. The system consists of two alternating components, a Tweet hijacking classification module and an update module. Our hijacking classification module is an ensemble of classifiers initially fit to our seed training set of 2770 Snorkel-labeled #MeToo Tweets. As new #MeToo Tweets are posted, we collect them using the Twitter streaming API, label them using the classification module, and re-fit the classifiers in the update module. We describe the two modules in detail in the rest of this section.

4.1 Hijacked Tweet Classification Module

The ensemble consists of five classifiers, each of which assigns a score between 0 (*valid*) and 1 (*hijacked*); the final predicted label is a weighted sum of these scores. We consider any Tweet with score greater than 0.8 to be *confidently* hijacked and any Tweet with score less than 0.3 to be *confidently* valid; these thresholds were tuned to maximize performance on the expert-labeled validation set.

4.1.1 Known Users Classifier

This classifier keeps two lists of users: a *blacklist* of known hijackers and a *whitelist* of trusted users. If a user has posted many hijacked Tweets, it is likely that they will do so again in the future; if a user has posted many genuine #MeToo Tweets, they are likely to continue doing so.

We use the same blacklist and whitelist definitions as Sedhai and Sun (2018): the blacklist consists of known hijackers who have posted more than 5 *hijacked* Tweets; the whitelist consists of trusted users who have never posted a hijacked Tweet and have posted at least eight *valid* Tweets. The lists are initially populated using our seed training set.

If a Tweet is posted by a user on the known hijackers blacklist, this classifier returns 1 (*hijacked*). It returns 0 (*valid*) if the user is on the trusted users whitelist and the Tweet does not contain any words from a hijacked word list; this condition prevents adversarial attacks by spammers who pretend to be legitimate users at first and post hijacked Tweets after achieving a spot on the whitelist (Yang et al., 2013). Finally, if a user is on neither blacklist nor whitelist, the classifier returns 0.5.

To generate the hijacked word list, we maintain two dictionaries: a hijacked dictionary and valid dictionary, where we store the counts of how often each word appears in hijacked and valid Tweets in our training data. We set a cutoff on the number of unique Tweets in which a word needs to appear to be included in these dictionaries: 5 and 8 for hijacked and valid, respectively. For each word w, we estimate the probability of w appearing in *hijacked* and *valid* Tweets:

$$C^b_{\text{hijack}}(w) = \text{count}^b_{\text{hijack}}(w) + \gamma C^{b-1}_{\text{hijack}}(w)$$
$$C^b_{\text{valid}}(w) = \text{count}^b_{\text{valid}}(w) + \gamma C^{b-1}_{\text{valid}}(w)$$

$$p^b_h(w) = \frac{C^b_{\text{hijack}}(w)}{C^b_{\text{hijack}}(w) + C^b_{\text{valid}}(w)}$$
$$p^b_v(w) = \frac{C^b_{\text{valid}}(w)}{C^b_{\text{hijack}}(w) + C^b_{\text{valid}}(w)}$$

$\gamma \in [0, 1]$ is a decay term that comes into effect during the batch update stage of our system (Section 4.2). When updating the hijacked word list for batch b, decay is applied to the accumulated counts C from batch $b-1$. If $p^b_h(w) > p^b_v(w)$, we add w to the hijacked word list; if $p^b_h(w) <= p^b_v(w)$, we remove w from the list, if necessary.

One concern that may arise with this classifier is that a user might be put on the blacklist and not be able to get off, even if their posting behavior changes later. While we did not find many examples of users so affected in our experiments, this issue could be addressed by adding some criteria for users to get off the blacklist. For example, if blacklisted user posts more than a certain number of Tweets that are classified as *valid* by the ensemble, they should be removed from the blacklist.

4.1.2 Tweet Text Classifier

This classifier uses TF-IDF features from the Tweet text. As a prepossessing step, we remove punctua-

tion, URLs, emojis, and stop words, and we lowercase and lemmatize the remaining words. We also replace some of the most commonly used abbreviations with full phrases (for example, replacing "ASAP" with "as soon as possible"). After preprocessing, we convert each Tweet into a vector of TF-IDF scores and fit a logistic regression model to label them. This is the only classifier in our ensemble that focuses on the Tweet text itself.

4.1.3 Social Classifier

This classifier focuses on how a Tweet and its user interact with other Tweets and users. We train a random forest model using features based on the Twitter spam analysis of Sedhai and Sun (2017):

- Number of users who follow the posting user.
- Number of users that the posting user follows.
- Whether or not the posting user is verified.
- Number of times the Tweet is retweeted.
- Number of times the Tweet is liked.
- Number of hashtags used in the Tweet.

This classifier uses the number of retweets and likes, which can vary greatly depending on how recently a Tweet was posted; a very new Tweet will have substantially lower values than the older Tweets in the seed training dataset. To address this issue, the number of likes and retweets are fetched again each time the update module runs.

4.1.4 User Profile Classifier

Sedhai and Sun (2017) argue that legitimate users are more likely to provide Twitter profile descriptions than spammers. Further, we hypothesize that, if a user is an active member of a hashtag activism movement, his or her profile description is more likely to be related to the movement. The user profile classifier labels Tweets that are posted by users with non-empty profile descriptions using a simple bag-of-words logistic regression model; for users without a profile description, this classifier simply labels the Tweet as *hijacked*.

4.1.5 Ensemble Voting

If a Tweet is labeled by the known users classifier, we consider it to be *confidently* labeled. Otherwise, we label it with the remaining classifiers, experimenting with three voting strategies:

- **Simple Average** returns the label corresponding to the average of the classifiers' scores.
- **Majority Vote** converts each classifier score into a binary label, *hijacked* or *valid*, and returns the majority label.
- **Stacking Meta-Learner** uses a gradient boosting meta-learner to weight the classifiers. If the weighted score is greater that 0.5, the Tweet is *hijacked*, and *valid* otherwise.

4.2 Batch Update Module

Since hijackers may adapt their strategies over time to fool the hijacking classification module, our system must adapt over time to correctly detect new hijacking cases. In the batch update module, we first select *confident* labels from among the system's predictions since the last batch update and add them to the training data. We then update the known users lists and retrain the Tweet text, social, and user profile classifiers, as well as the Stacking Meta-Learner. In the experiments below, we compare different sampling strategies for adding Tweets to the training data:

- **No Update** does not perform any updates and continues to use the seed-trained model.
- **Update All** adds all *confidently* labeled Tweets from the previous batch.
- **Update Equal** preserves class balance in the training set. If there are n *hijacked* and m *valid* Tweets, this strategy adds $\min(n, m)$ of each, selecting the most confident labels (ie. closer to 1 or 0) first.

5 Seed Model Results and Analysis

We report the results of our initial seed-trained model on the Expert Test set. While there aren't existing systems (Section 2) that are directly comparable with our framework, we use the closest, Jain et al. (2015), as a baseline; Van Dam and Tan (2016) focused on predicting whether a given trending hashtag was being hijacked, rather than detecting individual hijacked Tweets, and Virnami et al. (2017) required much larger amounts of hand-labeled training data than we have available, as well as non-generalizable domain knowledge, like dictionaries of related hashtags and URLs.

Jain et al. detected hijacking for general hashtags using an unsupervised approach. They used

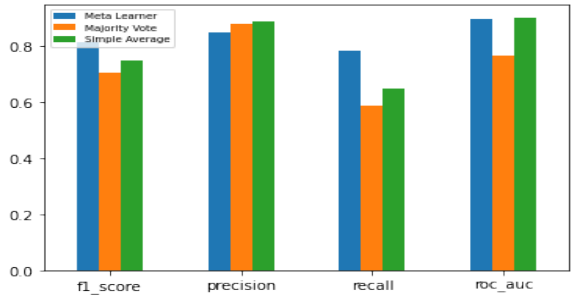


Figure 3: Comparison of ensemble voting strategies.

TF-IDF scores to create a dictionary of common words for each category of related hashtags, arguing that since hijacked Tweets are rare, they can be identified as having fewer words in common with their category. To use Jain et al.'s approach as a baseline, we use all 14,262 #MeToo Tweets that we collected, which we treat as a single category, and we collect an additional 500 Tweets for each month between October 2017 and November 2019 from each of Jain et al.'s categories (Table 3), totaling 13,000 Tweets per category.

We compare the performance of our seed-trained model with Jain et al. (2015), as well as random, majority, and minority baselines in Table 4. For the Stacking Meta-Learner, we report average scores across 100 runs. We see that our framework beats all baselines on all scores. The Stacking Meta-Learner outperforms each individual classifier on recall, while preserving relatively high precision, showing the importance of taking into account different aspects of a Tweet. Although the Tweet text classifier alone works well, the other classifiers boost the ensemble's performance on all metrics except precision, where it scores slightly lower. The Jain et al. baseline performs exactly the same as the minority baseline, labeling all Tweets as *hijacked*. This is likely due to hashtag activism Tweets using a more diverse vocabulary than general trending hashtags, resulting in Jain et al.'s TF-IDF dictionaries being less reliable for the #MeToo tweets.

5.1 Ensemble Voting Strategies

Figure 3 shows the performance of the three voting strategies using ROC-AUC score, precision, recall and F-measure as evaluation metrics. Both Simple Average and Stacking outperform Majority Vote. Simple Average and Stacking are very close, but we use Stacking in the rest of our experiments because it can adapt to changes in classifier performance over time by re-fitting the meta-learner at each batch update (Section 6).

5.2 Challenging Tweets

Figure 4 shows some Tweets that demonstrate why hijacking can be difficult to identify, even for human judges. Figure 4a could be considered spam, since they are promoting a product, but the product is related to #MeToo. Is "#MeToo merch" relevant to the social movement, or just taking advantage of it? This Tweet was labeled "hard to tell" by our expert annotators and omitted from the training set.

Figure 4b is an example of non-spam hijacking. This Tweet is about a different social movement targeting hunger in Sudan, and it hijacks several hashtags, including #MeToo. The Tweet uses language similar to that of social movement Tweets in general and was labeled *valid* by our system.

Figure 4c shows a joke Tweet from a user that exclusively posts off-color jokes and was added to the known hijackers blacklist during seed set training. However, this particular Tweet is arguably related to #MeToo, showing that even blacklisted users can occasionally post non-hijacked Tweets.

Finally, Figure 4d quotes a #MeToo-related Tweet, illustrating why we filter out Tweets that are replies to other Tweets. While this Tweet was correctly labeled as *valid* by our system, it would be impossible to tell that it is relevant without the quoted content; if it had been a reply instead of a quote, the required context would be missing.

6 Batch Update Results and Analysis

To evaluate how our framework performs over time, we collect all #MeToo Tweets posted from February to May 2020, totaling 122,792 *Live* Tweets, and use the batch update module to update the system every 24 hours: we use the previous model to label all Tweets posted in the next 24-hour window, update the training set with any new *confidently* labeled Tweets, and retrain the model. For evaluation, we sample 120 Live Tweets from each month[2], evenly split between predicted *hijacked* and predicted *valid* Tweets, and we use our expert annotators to obtain gold labels (Table 2).

Table 5 shows the performance of our ensemble using two different voting strategies, Stacking Meta-Learner and Simple Average, as well as the Tweet text classifier alone. We see that while the seed-trained Stacking Meta-Learner (top section)

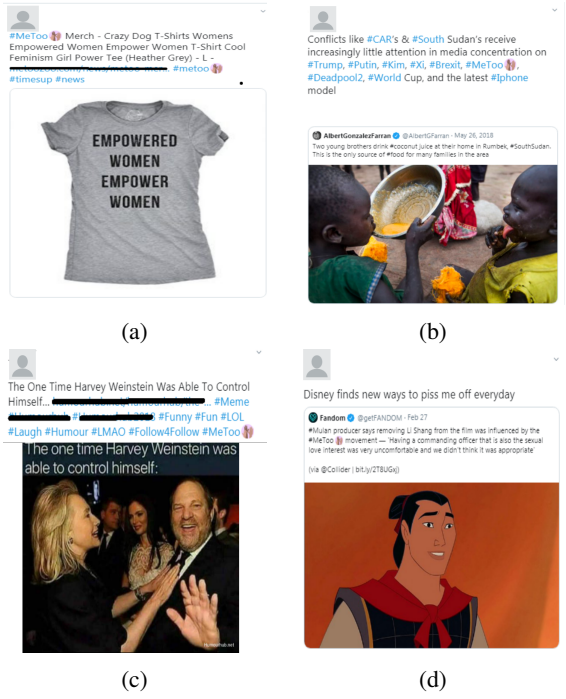
[2]We sample only 20 Tweets from April, as some days are missing due to an interruption in our collection script.

Category	Hashtags
Technology	#Android, #Apple, #Smartphone,#ios,#dell
Entertainment	#CSKvsMI, #Filmfare ,#MissWorld,#Maroon5,#Justin
Politics	#namo, #congress, #AAP, #BJP,# namobirthday
Brands	#puma, #adidas,#Samsung,#Lakme
Others	#happy, #Birthday, #Rain, #Sunny, #KillMe

Table 3: Hashtag categories used in the Jain et al. (2015).

Model	ROC-AUC	Precision	Recall	F-measure
Known User Classifier-BL	0.562	0.812	0.153	0.257
Known User Classifier-WL	0.519	**1.000**	0.038	0.074
Text Classifier	0.839	0.858	0.782	**0.818**
Social Classifiers	0.722	0.769	0.588	0.667
User Profile Classifier	0.666	0.760	0.447	0.563
Stacking Meta-Learner	**0.896**	0.847	0.784	0.814
Jain et al.	0.500	0.450	1.000	0.620
Random Baseline	0.514	0.463	0.518	0.489
Majority Baseline	0.500	0.000	0.000	0.000
Minority Baseline	0.500	0.450	**1.000**	0.620

Table 4: Experimental results for the seed-trained models. The top section shows the performance of individual classifiers, the middle shows the ensemble using the Stacking Meta-Learner, and the bottom shows baselines.



(a) (b) (c) (d)

Figure 4: Examples of challenging Tweets. Coarse language used in Tweet (c) is omitted.

and Simple Average (middle section) perform similarly on the Expert Test set (Figure 3), the updated Stacking Meta-Learner significantly outperforms updated Simple Average on the Expert Live Sample set. The bootstrapping approach used by the batch update module risks adding incorrect, noisy labels to the training set; the Stacking Meta-Learner has the advantage of being able to lower the weights of classifiers badly affected by such noise.

The seed-trained Tweet text classifier performs similarly to the seed Stacking Meta-Learner ensemble (Table 4). However, with the No Update strategy, the Tweet text classifier loses about 0.16 F-measure on the Expert Live Samples set compared to on the Expert Test set, while the Stacking Meta-Learner ensemble loses less than 0.1 F-measure, suggesting that the lexical features of the Tweet text classifier are more strongly affected by changes over time, while the other classifiers in the ensemble help mitigate this effect.

The Update All strategy also affects the Tweet text classifier much worse than it does the Stacking Meta-Learner. The seed training set is constructed to be balanced between *hijacked* and *valid* Tweets. However, *hijacked* Tweets are much rarer than *valid* Tweets "in the wild," and as the batch update module adds new Tweets to the training data, the *valid* Tweets quickly outnumber the *hijacked* Tweets. With this unbalanced training set, the Update All strategy results in very high precision and abysmally low recall for the Tweet text classifier. Again, the Stacking Meta-Learner ensemble is more robust; while its performance using the Update All strategy is worse than with No Update or Update Equal, it is not affected as strongly; it is able to lower the weights of classifiers, like Tweet text, that become less reliable as the training data grows imbalanced. Overall, the Update Equal

Model	ROC-AUC	Precision	Recall	F-measure
Stacking Meta-Learner with No Update	**0.764**	0.767	0.675	0.718
Stacking Meta-Learner with Update All	0.664	0.589	0.656	0.621
Stacking Meta-Learner with Update Equal	0.751	0.658	**0.801**	**0.722**
Simple Average with No Update	0.673	0.732	0.470	0.573
Simple Average with Update All	0.621	0.750	0.318	0.447
Simple Average with Update Equal	0.723	0.629	0.775	0.694
Text Classifier with No Update	0.727	0.806	0.550	0.654
Text Classifier with Update All	0.638	**0.885**	0.305	0.453
Text Classifier with Update Equal	0.759	0.856	0.589	0.698

Table 5: Experimental results using three different update strategies. The top section shows the performance of the ensemble using the Stacking Meta-Learner, the middle shows the ensemble using Simple Average voting, and the bottom section shows the Tweet text classifier trained alone, without the other classifiers.

strategy performs the best, adding an equal number of *hijacked* and *valid* Tweets at each batch update to preserve class balance in the training set.

Figure 5 shows Tweets that are labeled correctly by our Live Update system, but incorrectly by the seed-trained system. Figure 5a is correctly labeled as *hijacked* after Live Updates, while the seed system is mislead by the political hashtags. Figure 5b is correctly labeled as *valid* by the Live Update system, while the seed system labels it as *hijacked*, likely because of hashtags referring to actor Johnny Depp, coupled with the word "media."

(a)

(b)

Figure 5: Examples of Tweets labeled correctly by our Live Update system but not by the seed systems.

7 Conclusion

We have presented a weakly-supervised, bootstrapping framework to detect Tweet-level hashtag hijacking targeting social movements, using a combination of features based on the Tweet text, user profile, and other Tweet properties. We focus on the #MeToo movement, but our methodology can be applied to any movement or hashtag. Our approach is not limited to specific contexts and takes into account the changing characteristics of hashtag use over time. To best of our knowledge, this is the first time that a semi-supervised method is used to detect hashtag hijacking at the Tweet level.

Avenues for future work include addressing the class imbalance and error propagation that results in lower system performance over time, as well as exploring other types of classifiers. A potential solution to the error propagation problem may be to use active learning to obtain human-labeled samples at regular intervals to regulate our system. To reduce the expense of such annotation, submodular data subset selection can again be used to choose the most informative examples to label. Additional classifiers, such as one that scrapes linked webpages, or one that handles embedded images, could boost the overall performance of the ensemble.

We hope that this work encourages others to address the task of detecting Tweet-level hashtag hijacking and to develop other weakly-supervised approaches for Twitter data.

References

Leticia Bode, Alexander Hanna, JungHwan Yang, and Dhavan Shah. 2015. Candidate networks, citizen clusters, and political expression: Strategic hashtag use in the 2010 midterms. *The ANNALS of the American Academy of Political and Social Science*, 659:149–165.

Philipp Darius and Fabian Stephany. 2019. Twitter "hashjacked": Online polarisation strategies of germany's political far-right.

Marco Deseriis. 2017. Hacktivism: On the use of botnets in cyberattacks. *Theory, Culture & Society*, 34(4):131–152.

Hongyu Gao, Yan Chen, Kathy Lee, Diana Palsetia, and Alok N Choudhary. 2012. Towards online spam

filtering in social networks. In *NDSS*, volume 12, pages 1–16.

N. Hampson. 2012. Hacktivism: A new breed of protest in a networked world. *Boston College international and comparative law review*, 35:511.

Xia Hu, Jiliang Tang, and Huan Liu. 2014. Online social spammer detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Xia Hu, Jiliang Tang, Yanchao Zhang, and Huan Liu. 2013. Social spammer detection in microblogging. In *Twenty-third international joint conference on artificial intelligence*. Citeseer.

Nikita Jain, Pooja Agarwal, and Juhi Pruthi. 2015. Hashjacker — detection and analysis of hashtag hijacking on twitter. *International journal of computer applications*, 114(19).

Jan Kalbitzer, Thomas Mell, Felix Bermpohl, Michael Rapp, and Andreas Heinz. 2014. Twitter psychosis a rare variation or a distinct syndrome? *The Journal of nervous and mental disease*, 202:623.

Kyumin Lee, James Caverlee, and Steve Webb. 2010. Uncovering social spammers: social honeypots+ machine learning. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 435–442.

Simon Lindgren. 2019. Movement mobilization in the age of hashtag activism: Examining the challenge of noise, hate, and disengagement in the #metoo campaign. *Policy & Internet*, 11(4):418–438.

Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel. *Proceedings of the VLDB Endowment*, 11(3):269–282.

Jacob Schreiber, Jeffrey Bilmes, and William Stafford Noble. 2019. apricot: Submodular selection for data summarization in python.

S. Sedhai and A. Sun. 2018. Semi-supervised spam detection in twitter stream. *IEEE Transactions on Computational Social Systems*, 5(1):169–175.

Surendra Sedhai and Aixin Sun. 2017. An analysis of 14 million tweets on hashtag-oriented spamming*. *J. Assoc. Inf. Sci. Technol.*, 68(7):1638–1651.

Rukundo Solomon. 2017. Electronic protests: Hacktivism as a form of protest in uganda. *Computer Law Security Review*, 33(5):718–728.

Paul A. Taylor. 2005. From hackers to hacktivists: speed bumps on the global superhighway? *New Media & Society*, 7(5):625–646.

Courtland VanDam and Pang-Ning Tan. 2016. Detecting hashtag hijacking from twitter. In *Proceedings of the 8th ACM Conference on Web Science*, WebSci '16, pages 370–371, New York, NY, USA. ACM.

Deepali Virmani, Nikita Jain, Ketan Parikh, and Abhishek Srivastava. 2017. Hashminer: Feature characterisation and analysis of hashtag hijacking using real-time neural network. *Procedia Computer Science*, 115:786 – 793. 7th International Conference on Advances in Computing Communications, ICACC-2017, 22-24 August 2017, Cochin, India.

Alex Hai Wang. 2010. Don't follow me: Spam detection in twitter. In *2010 International Conference on Security and Cryptography (SECRYPT)*, pages 1–10.

Kai Wei, Yuzong Liu, Katrin Kirchhoff, Chris Bartels, and Jeff Bilmes. 2014. Submodular subset selection for large-scale speech training data. pages 3311–3315.

C. Yang, R. Harkreader, and G. Gu. 2013. Empirical evaluation and new design for fighting evolving twitter spammers. *IEEE Transactions on Information Forensics and Security*, 8(8):1280–1293.

Guobin Yang. 2016. Narrative agency in hashtag activism: The case of# blacklivesmatter. *Media and Communication*, 4(4):13.

Sarita Yardi, Daniel Romero, Grant Schoenebeck, and Danah Boyd. 2010. Detecting spam in a twitter network. *First Monday*, 15.

A Mechanical Turk Interface and Instructions

Figure 6 shows screenshots of the instructions and interface for the Human Intelligence Task we use to label the seed training set.

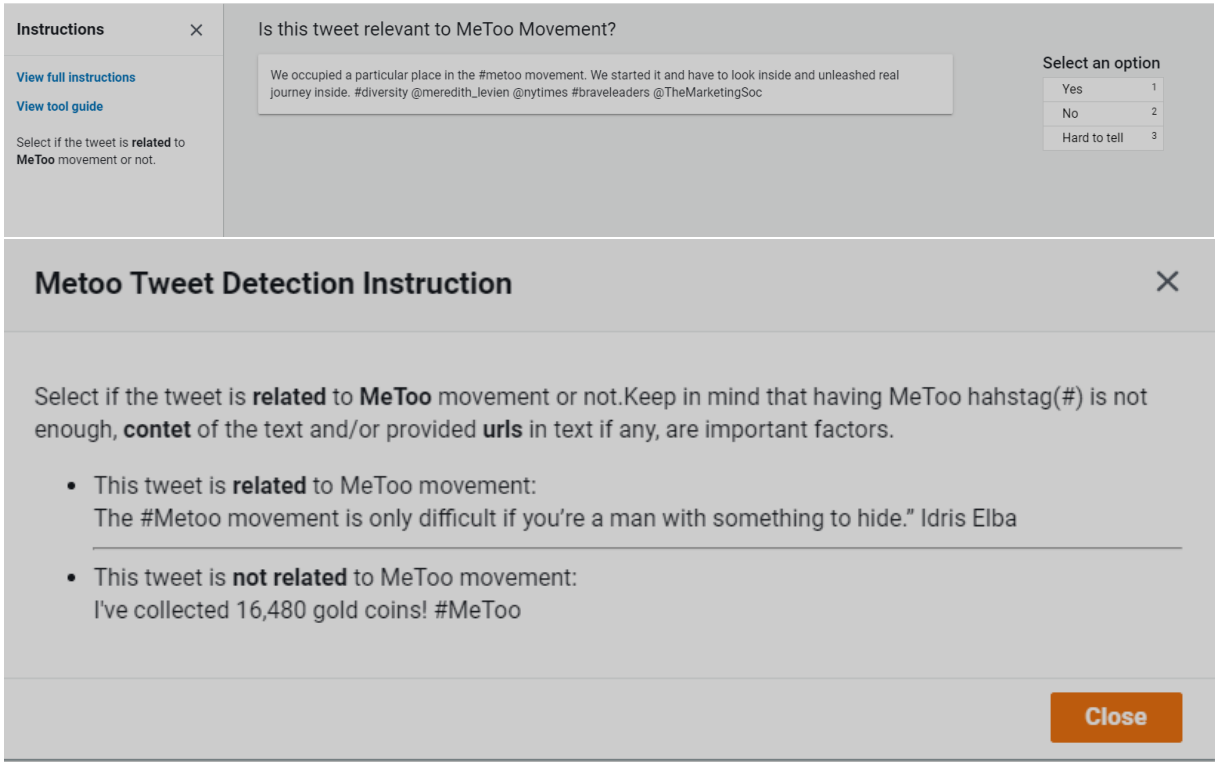


Figure 6: Our AMT task interface and instructions.