# The Fake News Vaccine

## A Content-Agnostic System for Preventing Fake News from Becoming Viral

Oana Balmau[3], Rachid Guerraoui[1], Anne-Marie Kermarrec[2], Alexandre Maurer[1(✉)], Matej Pavlovic[1], and Willy Zwaenepoel[3]

[1] École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
{rachid.guerraoui,alexandre.maurer,matej.pavlovic}@epfl.ch
[2] Mediego, Cesson-Sévigné, France
anne-marie.kermarrec@mediego.com
[3] University of Sydney, Camperdown, Australia
{oana.balmau,willy.zwaenepoel}@sydney.edu.au

**Abstract.** While spreading fake news is an old phenomenon, today social media enables misinformation to instantaneously reach millions of people. Content-based approaches to detect fake news, typically based on automatic text checking, are limited. It is indeed difficult to come up with general checking criteria. Moreover, once the criteria are known to an adversary, the checking can be easily bypassed. On the other hand, it is practically impossible for humans to check every news item, let alone preventing them from becoming viral.

We present CREDULIX, the first content-agnostic system to prevent fake news from going viral. CREDULIX is implemented as a *plugin* on top of a social media platform and acts as a *vaccine*. Human fact-checkers review a *small* number of popular news items, which helps us estimate the inclination of each user to share fake news. Using the resulting information, we automatically estimate the probability that an unchecked news item is fake. We use a *Bayesian* approach that resembles *Condorcet's Theorem* to compute this probability. We show how this computation can be performed in an *incremental*, and hence *fast* manner.

## 1 Introduction

The expression "fake news" has become very popular after the 2016 presidential election in the United States. Both political sides accused each other of spreading false information on social media, in order to influence public opinion. Fake news have also been involved in Brexit and seem to have played a crucial role in the French election. The phenomenon is considered by many as a threat to democracy, since the proportion of people getting their news from social media is significantly increasing.

We present CREDULIX, the first content-agnostic system to prevent fake news from getting *viral*. From a software perspective, CREDULIX is a plugin to a social media platform. From a more abstract perspective, it can also be viewed as a


M. F. Atig and A. A. Schwarzmann (Eds.): NETYS 2019, LNCS 11704, pp. 347–364, 2019.
https://doi.org/10.1007/978-3-030-31277-0_23

*vaccine* for the social network. Assuming the system has been exposed to some (small) amount of fake news in the *past*, CREDULIX enables it to prevent *future* fake news from becoming viral. It is important to note that our approach does not exclude other (e.g. content-based) approaches, but *complements* them.

CREDULIX retains the idea of using a team of certified human fact-checkers. However, we acknowledge that they cannot review *all* news items. Such an overwhelming task would possibly require even more fact-checkers than users. Here, the fact-checkers only check a *few viral* news items, i.e., ideally news items that have been shared and seen the most on the social network[1]. Many such fact-checking initiatives already exist all around the world (e.g., [2,3]). Such checks enable us to build *user credulity records*: records of which fact-checked items a user has seen and shared.

We use a *Naive Bayes* approach[2] to estimate the credibility of news items based on which users shared them and how these users treated fake news in the past. News items considered fake with a sufficiently high probability can then be prevented from further dissemination, i.e., from becoming viral. Our result is in the spirit of *Condorcet's jury Theorem* [15], which states that a very high level of reliability can be achieved by a large number of weakly reliable individuals. To determine the probability of falsehood of a news item X, we look at the behavior of users towards X. This particular behavior had a certain a priori probability to happen. We compute this probability based on the previously constructed user credulity records. Then, after determining the average fraction of fake news on the social network, we apply Laplace's Rule of Succession [35] and then Bayes' Theorem [25] to obtain the desired probability.

When this probability goes beyond a threshold (say 99.9999%), the social network can react accordingly. E.g., it may stop showing the news item in other users' news feeds. It is important to note that our approach does not require any users to share a large amount of fake news. It suffices that some users share more fake news than others.

Also note that CREDULIX does not completely eliminate the spreading of all fake news—unpopular items only seen by very few users might pass undetected, since it is the user interactions with an item that contribute to its detection. However, CREDULIX does prevent fake items from going viral—once a sufficient (but still low) number of users were exposed to a fake item, CREDULIX takes the

[1] Some news items are indeed seen by millions, and are easy to check a posteriori. For instance, according to CNN [1], the following fake news items were read by millions: *"Thousands of fraudulent ballots for Clinton uncovered"*; *"Elizabeth Warren endorsed Bernie Sanders"*; *"The NBA cancels 2017 All-Star Game in North Carolina"*.

[2] Naive Bayes approaches [32,36] assume that the random variables are independent, even if they are not totally independent in practice. This enables to simplify a problem that, otherwise, would be far too complex to tackle. Naive Bayes approaches work surprisingly well in many complex real-world situations, and are also very robust [32] ([36] explains some possible theoretical reasons for this). Here, the imprecision of the probability we compute is compensated by the fact that we choose a threshold which is extremely close to 1 (i.e., $1-10^{-6}$, or 99.9999%). Thus, even with an error of $\times 100$, the actual probability would be $1-10^{-4}$, with does not change much from our perspective.

appropriate action. Thus, fake news items that would otherwise become popular (arguably the most harmful ones) are always detected by CREDULIX.

Our approach is *generic* in the sense that it does not depend on any specific criteria. Here, for instance, we look at what users *share* to determine if a news item is *fake*. However, the approach is independent of the precise meanings of "share" and "fake": they could respectively be replaced by ("like" or "report") and ("funny", "offensive", "politically liberal" or "politically conservative").

In some sense, CREDULIX shares similarities with recommender systems, since the news items are being classified into categories based on users' reactions. However, the purpose of CREDULIX is fundamentally different from that of recommender systems. While recommenders aim to provide personalized content based on users' preferences, CREDULIX' classification of news items being true or fake is independent of the requesting user. The aim is to prevent the spread of fake news, not to provide users with a personalized selection of news articles.

Turning the theory behind CREDULIX into a system deployable in practice is a non-trivial task. In this paper we address these challenges as well. In particular, we present a practical approach to computing news item credibility in a *fast*, incremental manner.

We implement CREDULIX as a standalone Java plugin and connect it to Twissandra [14] (an open source Twitter clone), which serves as a baseline system. CREDULIX interferes very little with the critical path of users' operations and thus has a minimal impact on user request latency. We evaluate CREDULIX in terms of its capacity to detect fake news as well as its performance overhead when applied to a real social network of over 41M users [27]. After fact-checking the 1024 most popular news items (out of a total of over 35M items), over 99% of unchecked fake news items are correctly detected by CREDULIX. We also show that CREDULIX does not incur significant overhead in terms of throughput and latency of user operations (sharing items and viewing the news feed): average latency increases by at most 5%, while average throughput decreases by at most 8%.

## 2 Theoretical Foundations

In this section we give an intuition of the theoretical result underlying CREDULIX, followed by its formalization as a theorem. We finally show how to restate the problem in a way that allows efficient, fast computation of news item credibility.

### 2.1 Intuition

The context is a social network where users can post news items, such as links to newspapers or blog articles. Users exposed to these news items can in turn share them with other users (Twitter followers, Facebook friends etc.). The social network has a fact-checking team

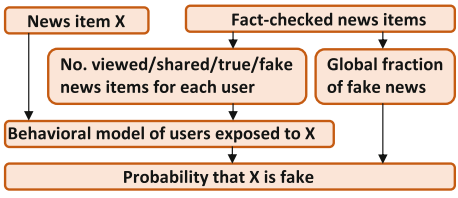


**Fig. 1.** News item falsehood probability computation.

whose role is to determine whether certain news items are fake (according to some definition of fake)[3]. The news items that the fact-checking team needs to check is very low compared to the total number of items in the social network.

**The Main Steps.** Our approach goes through the following three main steps:

1. The fact-checking team reviews few news items (ideally those that have been the most viral ones in the past). This is considered the ground truth in our context.
2. CREDULIX creates a probabilistic model of each user's sharing behavior based on their reactions (share/not share) to the fact-checked items in Step (1). This captures the likelihood of a user to share true (resp. fake) items in the future.
3. For a new, unchecked news item $X$, we use the behavior models generated in Step (2) to determine the probability that $X$ is fake, based on who viewed and shared $X$.

A high-level view of our technique is depicted in Fig. 1. We use a Bayesian approach. For example, if an item is mostly shared by users with high estimated probabilities of sharing fake items, while users with high estimated probabilities of sharing true items rarely share it, we consider the item likely to be fake (see Fig. 2).

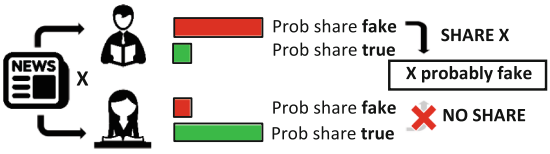


**Fig. 2.** Users reacting to new item $X$.

The above-mentioned main steps happen *continuously and in parallel*, as the system is running. Over time, fresh news items are created, new users join the system and users interact with news items. Step (1) happens periodically, as the fact-checking team reviews more articles and increases the number of news articles that are part of the ground truth. Step (2) and (3) happen continuously, as users are interacting with news items. Updates in users' sharing behavior (Step (2)) trigger updates in the likelihood of an unchecked news item being classified as true or fake (Step (3)).

In order to initialize CREDULIX, the fact-checking team needs to review a low number of news items. The required number of reviewed items depends on their relative popularity, but not on the overall number of items in the system. In our evaluation, reviewing 1024 most popular news items is sufficient for fast and accurate fake news detection. To work with real-world item popularity, we use a corpus of 35M real tweets to select our items. Regardless of the overall number of items, we would still only need to review 1024 of them. We consider that this number of items is still easy to fact-check through non-automated techniques, e.g. by making use of existing fact-checking initiatives [2–6].

[3] Our truth and falsehood criteria here are as good as the fact-checking team. CREDULIX trusts the fact-checking team to correctly identify true and false news items.

**Preventing the Spread of Fake News.** Once we estimate the probability of a news item $X$ being fake, preventing its spread becomes easy. Let $p_0$ be any cutoff probability threshold. Each time a user views or shares $X$, we compute $p$, the probability of $X$ being fake, which we detail below. If $p \geq p_0$ (i.e., $X$ has a probability at least $p_0$ to be fake), CREDULIX stops showing $X$ in the news feed of users, preventing $X$ from spreading and becoming viral.

### 2.2 Basic Fake News Detection

**User Behavior.** We model the behavior of a user $u$ using the two following probabilities:

- $P_T(u)$: probability that $u$ shares a news item if the item is true.
- $P_F(u)$: probability that $u$ shares a news item if the item is fake.

The probabilities $P_T(u)$ and $P_F(u)$ are assumed to be independent between users. In practice, this is the case if the decision to share a news item $X$ is mainly determined by $X$ itself (Fig. 3).

We obtain estimates of $P_T(u)$ and $P_F(u)$ for each user based on the user's behavior (share/not share) with respect to fact-checked items. For any given user $u$, let $v_T(u)$ (resp. $s_T(u)$) denote the number of fact-checked true news items *viewed* (resp. *shared*) by $u$, and $v_F(u)$ (resp. $s_F(u)$) the number of fact-checked fake news items *viewed* (resp. *shared*) by $u$. We call the tuple $(v_T(u), s_T(u), v_F(u), s_F(u))$ the *User Credulity Record* (UCR) of $u$.

| | |
|---|---|
| $v_T(u)$ | Number of fact-checked *true* items *viewed* by $u$ |
| $s_T(u)$ | Number of fact-checked *true* items *shared* by $u$ |
| $v_F(u)$ | Number of fact-checked *false* items *viewed* by $u$ |
| $s_F(u)$ | Number of fact-checked *false* items *shared* by $u$ |

**Fig. 3.** User Credulity Record (UCR).

**Probability of a News Item Being Fake.** Let $X$ be a news item (not fact-checked). Let $V$ and $S$ be any two sets of users that have *viewed* and *shared* $X$, respectively. In the following, we define a function $p(V, S)$. We then show (Theorem 1) that $p(V, S)$ is the probability that $X$ is fake.

Let $g$ be the estimated *global fraction* of fake news items in the social network, with $g \in (0, 1)$. The fraction $g$ can be estimated by fact-checking a set of news items picked uniformly at random from the whole social network.

We now define the 6 following functions (based on the UCR). Note that these functions do not correspond to anything in particular. We only use them as intermediary steps to simplify notation when defining define $p(V, S)$, and to make the following proofs simpler.

- $\beta_1(u) = (s_T(u) + 1)/(v_T(u) + 2)$
- $\beta_2(u) = (s_F(u) + 1)/(v_F(u) + 2)$
- $\beta_3(u) = (v_T(u) - s_T(u) + 1)/(v_T(u) + 2)$
- $\beta_4(u) = (v_F(u) - s_F(u) + 1)/(v_F(u) + 2)$
- $\pi_T(V, S) = \prod_{u \in S} \beta_1(u) \prod_{u \in V-S} \beta_3(u)$
- $\pi_F(V, S) = \prod_{u \in S} \beta_2(u) \prod_{u \in V-S} \beta_4(u)$

We now define $p(V, S)$ as follows (at this point, this is just an arbitrary definition; in Theorem 1, we show that $p(V, S)$ actually is the probability that $X$ is fake):

$$\boxed{p(V, S) = \frac{g\pi_F(V, S)}{g\pi_F(V, S) + (1 - g)\pi_T(V, S)}} \tag{1}$$

Let $g^*$ be the real fraction of fake news items in the social network (of which $g$ is an estimate). We first show that, if $g = g^*$, then the probability that $X$ is fake is $p(V, S)$ (Theorem 1). Then, we consider the case where we can only assume that $g \leq g^*$ (i.e. we only have a lower bound of $g^*$). We then show that the probability that $X$ is fake is at least $p(V, S)$, which gives us a conservative estimate (Theorem 2).

**Theorem 1.** *Let $g = g^*$. A news item* viewed *by a set of users $V$ and* shared *by a set of users $S$ is fake with probability $p(V, S)$.*

*Proof.* According to Laplace's Rule of Succession [35], we have $P_T(u) = \beta_1(u)$ and $P_F(u) = \beta_2(u)$. Consider a news item $X$ that has not been fact-checked. Consider the following events: (1) $E$: $X$ viewed by a set of users $V$ and shared by a set of users $S$; (2) $F$: $X$ is fake; (3) $T$: $X$ is true. Our goal is to evaluate $P(F|E)$: the probability that $X$ is fake knowing $E$.

- If $X$ is true, $P(E|T) = \prod_{u \in S} P_T(u) \prod_{u \in V-S}(1 - P_T(u)) = \pi_T(V, S)$.
- If $X$ is fake, $P(E|F) = \prod_{u \in S} P_F(u) \prod_{u \in V-S}(1 - P_F(u)) = \pi_F(V, S)$.

The probability that $X$ is fake (independently of $E$) is $P(F) = g^*$, and the probability that $X$ is true (independently of $E$) is $P(T) = 1 - g^*$. Thus, we can determine the probability that $E$ is true: $P(E) = P(E|T)P(T) + P(E|F)P(F) = (1 - g^*)\pi_T(V, S) + g^*\pi_F(V, S)$. $P(F|E) = P(E|F)P(F)/P(E)$ according to Bayes' Theorem [25]. Then, $P(F|E) = g^*\pi_F(V, S)/(g^*\pi_F(V, S) + (1 - g^*)\pi_T(V, S)) = p(V, S)$. Thus, the result.

If $g^*$ is unknown, we assume that $g$ is a lower bound of $g^*$. We get in this case the following theorem.

**Theorem 2.** *For $g \leq g^*$, a news item* viewed *by a set of users $V$ and* shared *by a set of users $S$ is fake with probability* at least *$p(V, S)$.*

*Proof.* First, note that $\pi_T(V, S)$ and $\pi_F(V, S)$ are strictly positive by definition. Thus, the ratio $\pi_T(V, S)/\pi_F(V, S)$ is always strictly positive. $\forall x \in (0, 1)$, let $g(x) = x\pi_F(V, S)/(x\pi_F(V, S) + (1 - x)\pi_T(V, S))$. Then, $p(V, S) = h(g)$, and according to Theorem 1, the news item is fake with probability $h(g^*)$. Written differently, $g(x) = 1/(1 + k(x))$, with $k(x) = (1/x - 1)\pi_T(V, S)/\pi_F(V, S)$. As $g \leq g^*$, $1/g \geq 1/g^*$, $1 + k(g) \geq 1 + k(g^*)$ and $h(g) \leq h(g^*)$. Thus, the result.

### 2.3 Fast Fake News Detection

Credulix' measure of credibility of a news item $X$ is the probability $p(V, S)$ that $X$ is fake. An obvious way to compute this probability is to recalculate $p(V, S)$ using Eq. (1) each time $X$ is viewed or shared by a user. Doing so, however, would be very expensive in terms of computation. Below, we show an efficient method for computing news item credibility. We first describe the computation of UCRs, and then present our fast, incremental approach for computing news item credibility using *item ratings* and *UCR scores*. This is crucial for efficiently running Credulix in practice.

**Computing User Credulity Records (UCRs).** Recall that the four values $(v_T(u), s_T(u), v_F(u), s_F(u))$ constituting a UCR only concern *fact-checked* news items. We thus update the UCR of user $u$ (increment one of these four values) in the following two scenarios.

1. When $u$ views or shares a news item that has been fact-checked (i.e., is known to be true or fake).
2. Upon fact-checking a news item that $u$ had been exposed to.

In general, the more fact-checked news items a user $u$ has seen and shared, the more *meaningful* $u$'s UCR. Users who have not been exposed to any fact-checked items cannot contribute to Credulix.

**Item Rating.** In addition to $p(V, S)$, we introduce another measure of how confident Credulix is about $X$ being fake: the *item rating* $\alpha(V, S)$, whose role is equivalent to that of $p(V, S)$. We define it as $\alpha(V, S) = \pi_T(V, S)/\pi_F(V, S)$, $V$ and $S$ being the sets of users that viewed and shared $X$, respectively. If we also define $\alpha_0 = (1/p_0 - 1)/(1/g - 1)$ as the rating threshold corresponding to the probability threshold $p_0$, then, $p(V, S) \geq p_0$ is equivalent to $\alpha(V, S) \leq \alpha_0$.

We have $p(V, S) = g\pi_F(V, S)/(g\pi_F(V, S) + (1 - g)\pi_T(V, S)) = g/(g + (1 - g)(\pi_T(V, S)/\pi_F(V, S))) = g/(g + (1 - g)\alpha(V, S))$. We have $p(V, S) \geq p_0$ if and only if $g/p(V, S) \leq g/p_0$, that is: $g + (1 - g)\alpha(V, S) \leq g/p_0$, which is equivalent to $\alpha(V, S) \leq (1/p_0 - 1)/(1/g - 1)$, that is: $\alpha(V, S) \leq \alpha_0$.

When the item $X$ with $\alpha(V, S) \leq \alpha_0$ is about to be displayed in a user's news feed, Credulix suppresses $X$. Note that $\alpha_0$ can be a fixed constant used throughout the system, but may also be part of the account settings of each user, giving users the ability to control how "confident" the system needs to be about the falsehood of an item before suppressing it.

According to the definition of $\pi_T(V, S)$ and $\pi_F(V, S)$, each time $X$ is viewed (resp. shared) by a new user $u$, we can update $X$'s rating $\alpha(V, S)$ by multiplying it by $\gamma_v(u) = \beta_1(u)/\beta_2(u)$ (resp. $\gamma_s(u) = \beta_3(u)/\beta_4(u)$). We call $\gamma_v(u)$ and $\gamma_s(u)$ respectively the *view score* and *share score* of $u$'s UCR, as their value only depends on $u$'s UCR. Consequently, when a user views or shares $X$, we only need to access a single UCR in order to update the rating of $X$. This is what allows Credulix to update news item credibility fast, without recomputing Eq. (1) each time the item is seen by a user.

In what follows, we refer to $\gamma_v(u)$ and $\gamma_s(u)$ as $u$'s *UCR score*. The more a UCR score differs from 1, the stronger its influence on an item rating (which is computed as a product of UCR scores). We consider a UCR score to be *useful* if it is different from 1 (as item ratings are products of UCR scores).

## 3 Credulix as a Social Media Plugin

Credulix can be seen as a plugin to an existing social network, like, for instance, Facebook's translation feature. The translator observes the content displayed to users, translating it from one language to another. Similarly, Credulix observes news items about to be displayed to users and tags or suppresses those considered fake.

Despite the fast computation described in Sect. 2.3, there are still notable challenges posed by turning an algorithm into a practical system. In order for the Credulix plugin to be usable in practice, it must not impair user experience. In particular, its impact on the latency and throughput of user operations (retrieving news feeds or tweeting/sharing articles) must be small. Our design is motivated by minimizing Credulix' system resource overhead.

**Selective Item Tracking.** Every second, approximately 6000 new tweets appear on Twitter and 50000 new posts are created on Facebook [7]. Monitoring the credibility of all these items would pose significant resource overhead. With Credulix, each view/share event requires an additional update to the news item's metadata. However, we do not need to keep track of all the items in the system, but just the ones that show a potential of becoming viral.

Credulix requires each item's metadata to contain an additional bit indicating whether that item is *tracked*. The rating of item $X$ is only computed and kept up to date by Credulix if $X$ is tracked.

We set the *tracked* bit for item $X$ when $X$ is shared by an *influential user*. We define influential users as users who have a high number of followers. The intuition behind this approach is that a news item is more likely to become viral if it is disseminated by a well-connected user [23]. The follower threshold necessary for a user to be considered influential is a system parameter.

**Interaction with the Social Media Platform.** We consider two basic operations a user $u$ can perform:

- *Sharing* a news item and
- *Viewing* her own news feed.

*Sharing* is the operation of disseminating a news item to all of $u$'s followers (e.g., tweeting, sharing, updating Facebook status etc.). *Viewing* is the action of requesting the news feed, to see new posts shared by users that $u$ follows.

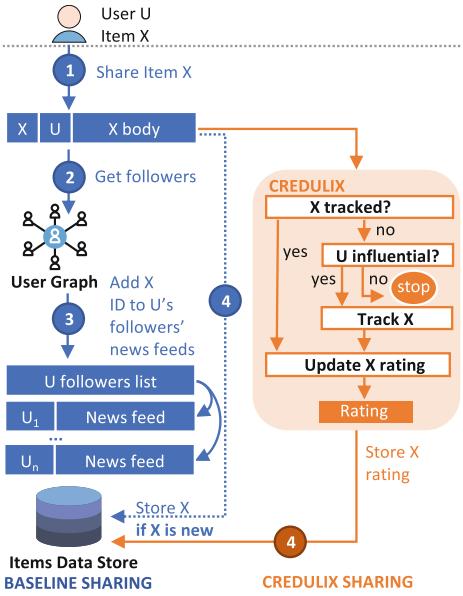


**Fig. 4.** Sharing a news item

**Baseline Sharing.** A schema of the *Share* operation is shown in Fig. 4. The regular flow of the operation is shown in blue and CREDULIX is shown in orange. User $u$ shares an item $X$ (1). First, the social graph is queried to retrieve $u$'s followers (2). The system then appends the ID of $X$ to the news feeds of $u$'s followers (3). Finally, if $X$ is a new item, the body of $X$ is stored in a data store (4).

**Sharing with Credulix.** If $u$ is not an influential user, the flow of the share operation described above stays the same. If $u$ is influential, we mark $X$ as tracked and associate an item rating with $X$, because we expect $X$ to potentially become viral. If $X$ is tracked, CREDULIX updates the rating of $X$ using $u$'s UCR share score. Thus, for tracked items, CREDULIX may require one additional write to the data store compared to the Baseline version, in order to store the updated item rating. This is done off the critical path of the user request, hence not affecting request latency.

**Baseline News Feed Viewing.** A schema of the *View* operation is shown in Fig. 5. User $u$ requests her news feed (1). For each item ID in $u$'s news feed (stored in memory), the system retrieves the corresponding item body from the data store (2) and sends all of them back to the user (3).

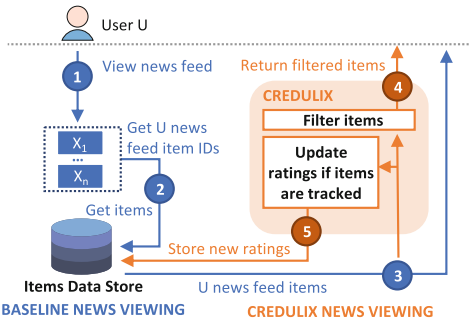


**Fig. 5.** Viewing news feed

**Viewing News Feed with Credulix.** CREDULIX augments the *View* operation in two ways.

First, after the news feed articles are retrieved from the data store, CREDULIX checks the ratings of the items, filtering out the items with a high probability of being fake. Second, if $u$'s news feed contains tracked items, CREDULIX updates the rating of those items using $u$'s UCR view score. Hence, a supplementary write to the data store is necessary, compared to the Baseline version, for storing the items' updated ratings. Again, we do this in the background, not impacting user request latency.

## 4 Evaluation

In this section, we evaluate our implementation of CREDULIX as a stand-alone Java plugin. We implement a Twitter clone where the *share* and *view* operation executions are depicted in Figs. 4 and 5. We refer to the Twitter clone as *Baseline* and we compare it to the variant with CREDULIX plugged in, which we call CREDULIX. For the data store of the Baseline we use Twissandra's data store [14], running Cassandra version 2.2.9 [11]. The main results of our evaluation are the following.

1. CREDULIX efficiently stops the majority of fake news from becoming viral, with no false positives. CREDULIX reduces the number of times a viral fake news item is viewed from hundreds of millions to hundreds of thousands (in Sect. 4.2).
2. CREDULIX' impact on system performance is negligible for both throughput and latency (in Sect. 4.3).

### 4.1 Experimental Setup and Methodology

We perform our evaluation using a real Twitter graph of over 41M users [27]. We consider users to be influential if they are among the 5% most followed users.

We use a set of tweets obtained by crawling Twitter to get a distribution of item popularity. Out of over 35M tweets we crawled, the 1024 (0.003%) most popular tweets are retweeted almost 90 million times, which corresponds to over 16% of all the retweets. Two key values influence CREDULIX' behavior:

- `r`**:** The number of fact-checked news items during UCR creation (i.e., the number of news items constituting the ground truth). We use `r` = 1024, which causes one third of the user population to have useful UCR scores.
- `msp`**:** The max share probability models users' intrinsic sharing behavior: how likely users are to share news items they are exposed to. This models how users react to news items. It is *not* a system parameter of CREDULIX. We expect `msp` to be different for different news items, as some items become viral (high `msp`) and some do not (low `msp`). Since we do not know the real-world value of `msp`, we do use it as a parameter of our experiments (not a parameter of CREDULIX itself), in order to explore the behavior of our system in all situations. As our evaluation shows, regardless of what the real value of `msp` is, CREDULIX effectively prevents fake items from going viral.

While the network and the tweets come from real data sets, we generate the user behavior (i.e., probability to share fake and true news items), as we explain below. We proceed in two steps:

1. *UCR creation*: determining the UCR (i.e., $v_T, s_T, v_F, s_F$) for each user based on propagation of fact-checked news items.
2. *Fake item detection*: using the UCRs obtained in the previous step, we use CREDULIX to detect fake news items and stop them from spreading.

This separation is only conceptual, for clarity of the presentation. As CREDULIX is running in a social network, both UCR creation (or UCR updates) and fake item detection happen *continuously and in parallel*.

**UCR Creation.** For each user $u$, we set $P_T(u)$ and $P_F(u)$ (see Sect. 2) to values chosen uniformly at random between 0 and `msp`. The likelihood of a user to share true or fake news is the main user characteristic used by CREDULIX.

We take a subset of `r` popular tweets from our tweet dataset and consider this subset the ground truth, randomly assigning truth values to items. This phase of our experiments simulates the human fact-checking process.

In practice, the true news items on real social media still greatly outnumber the fake news items. Thus, it might look intuitive to make the ratio between fake and true items generated this way correspond to the (rather small) fraction of fake items present in real social networks. However, contrary to the intuition, this is not necessary. CREDULIX works best if the ratio between true and fake items that constitute the ground truth is balanced, i.e., the fraction of fake news items is one half.[4] To achieve balance in the ground truth, it suffices to bias the fact-checking process towards items that are likely to be fake. Indeed, even many of the already existing fact-checking initiatives [2–6] tend to focus on fake items and their output is balanced, if not biased towards fake items. In order to stay conservative in our evaluation, we set this ratio to 1/4, meaning that each item in our generated ground truth is fake with probability 1/4 and true with probability 3/4.

To create the UCRs, we propagate these `r` ground-truth items through the social graph. We assign each of the `r` items a target share count, which corresponds to its number of retweets in our dataset. The propagation proceeds by exposing a random user $u$ to the propagated item $X$ and having $u$ decide (based on $P_T(u)$ and $P_F(u)$) whether to share $X$ or not. If $u$ shares $X$, we show $X$ to all $u$'s followers that have not yet seen it. During item propagation, we keep track of how many true/fake items each user has seen/shared and update the UCRs accordingly.

We repeat this process until one of the following conditions is fulfilled: (1) The target number of shares is reached, or (2) At least 80% of users have been exposed to $X$, at which point we consider the network saturated.

[4] One half as the optimal fraction of fake items in the ground truth is confirmed by our experiments.

At the end of the UCR creation step, each user has an updated UCR, tracking how many true/false items that user has seen/shared. These UCRs are used in the next phase for detecting which (not fact-checked) news items are fake.

**Fake Item Detection.** After creating the UCRs, we measure how effectively these can be leveraged to detect fake news. To this end, in the second step of the evaluation, we propagate news items through the social graph. We consider these items not fact-checked. One such experiment consists of injecting an item in the system, by making a random user $u$ share it. The propagation happens as in the previous phase, with two important differences. First, we do not update $u$'s UCR. Instead, whenever $u$ is exposed to an item, we update that item's rating using $u$'s UCR score. We use the share score if $u$ shares the item, otherwise we use the view score (see Sect. 2). Second, we only propagate the item once, continuing until the propagation stops naturally, or until the probability of an item being fake reaches $p_0 = 0.999999$.

In the evaluation, we are interested in whether (and how fast) CREDULIX reacts to fake items, and whether the propagation of true items stays unaffected. To this end, we repeat this experiment 500 times with a fake news item and 500 times with a true news item to obtain the results presented later in this section.

We conduct the experiments on a 48-core machine, with four 12-core Intel Xeon E7-4830 v3 processors operating at 2.1 GHz, 512 GB of RAM, running Ubuntu 16.04.

### 4.2 Stopping Fake News from Becoming Viral

This experiment presents the end-to-end impact of CREDULIX on the number of times users are exposed to fake news. To this end, we measure the number of times a user is exposed to a fake news item in the baseline case and compare it to a case with CREDULIX in place.

Figure 6 conveys results for items with varying rates of virality, modeled by our `msp` parameter. It shows how many times a user has been exposed to a fake item, cumulatively over the total number of fake items that we disseminate.

We can see that regardless of how viral the items would naturally become, CREDULIX is able to timely detect the fake items before they spread to too many users. CREDULIX restricts the number of views from hundreds of millions to tens or hundreds of thousands.

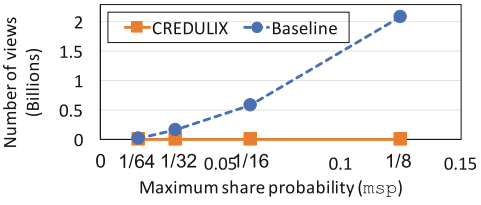


**Fig. 6.** Fake news spreading as a function of `msp` (lower is better). For low `msp`, news items do not become viral. For high `msp`, CREDULIX blocks the majority of fake news items.

None of our experiments encountered false positives (i.e., true items being incorrectly labeled as fake). Considering the increasing responsibility being attributed to social

network providers as mediators of information, it is crucial that true news items are not accidentally marked as fake.

In Fig. 7 we plot the percentage of fake items displayed with CREDULIX for two graph sizes. On a smaller graph of 1M users generated with the SNAP generator [28], CREDULIX achieves a lower fake item detection rate. This is because the impact of fact-checked items is smaller on a small graph, leading to fewer users with relevant UCR scores. This result suggests that on a real social graph that is larger than the one we use, CREDULIX would be more efficient than in our experiments.

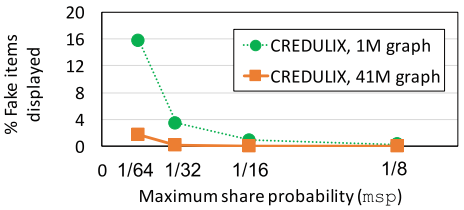


**Fig. 7.** Fake news spreading with CREDULIX, as a function of `msp`, for different social graph sizes (lower is better).

Figure 7 also shows how the detection rate depends on the tendency of users to share news items. The more viral the items get (the higher the `msp` value), the more effective CREDULIX becomes at fake item detection. Intuitively, the more items users share, the more precisely we are able to estimate their sharing behavior. The lower detection rate for small `msp` values does not pose a problem in practice, as a low `msp` also means that items naturally do not become viral (Table 1).

**Table 1.** Workload characteristics. The only parameter we vary is `msp`, from which the view/share ratio follows.

| `msp` value | % views, % shares |
|---|---|
| 1/8 | 94% views, 6% shares |
| 1/16 | 97% views, 3% shares |
| 1/32 | 99% views, 1% shares |
| 1/64 | 99.9% views, 0.1% Shares |

While not visible in the plot, it is worth noting that not only the relative amount of viewed fake items decreases, but also the *absolute* one. For example, while for `msp` = 1/32 a fake news item has been displayed almost 3k times (out of over 84k for Baseline), for `msp` = 1/16 a fake item has only been displayed 1.2k times (out of over 128k Baseline) in the 1M graph.

Interestingly, with increasing tendency of items to go viral (i.e. increasing `msp`), even the *absolute* number of displayed fake items decreases. The relative decrease effect is not due to an absolute increase for Baseline. Instead, it is due to a higher `msp` value ensuring more spreading of (both true and fake) news items in our UCR creation phase. This in turn produces better UCRs, increasing CREDULIX' effectiveness.

### 4.3 Credulix Overhead

We evaluate CREDULIX' impact on user operations' (viewing and sharing) *throughput* and *latency*. We present our results for four workloads, each corresponding to a value of `msp` used above (Fig. 6). Note that the `msp` values (1/8, 1/16, 1/32, and 1/64) also determine the ratio of view operations (respectively ca. 94%, 97%, 99%, and 99.9%). We present results for two social graph sizes:

41M users, and 1M users, with 16 worker threads serving user operations, showing that the CREDULIX' overhead in terms of throughput and latency is low.

Figure 8 shows the throughput comparison between CREDULIX and Baseline, for the four workloads. The throughput penalty caused by CREDULIX is at most 8%. The impact on throughput is predominantly caused by CREDULIX' background tasks, as detailed in Sect. 3. Moreover, CREDULIX does not add significant overhead relative to the Baseline as the graph size increases. The throughput differences between the two graph sizes are not larger than 10%. This is due to our design which relies on selective item tracking.

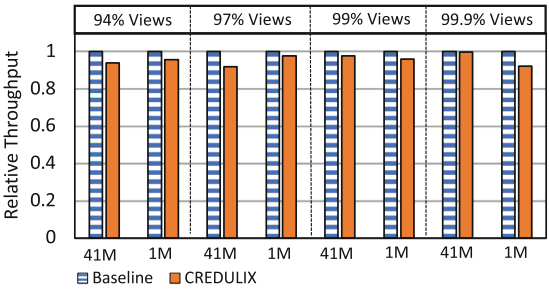


**Fig. 8.** CREDULIX' throughput overhead

Figure 9 shows view and share latencies for the 99.9% views workload. The latency values for the other workloads are similar and we omit them for brevity. For the 41M User Twitter graph, the average and $90^{th}$ percentile latencies are roughly the same for CREDULIX and for Baseline. We notice, however, heavier fluctuations for the 1M User graph. Overall, latency increases by at most 17%, at the $90^{th}$ percentile, while the median latency is the same for both operations, for both systems (4 ms per operation). The low overhead in latency is due to CREDULIX keeping its computation outside the critical path. Standard deviation of latencies is high both for the Baseline and for CREDULIX, for both share and view operations.

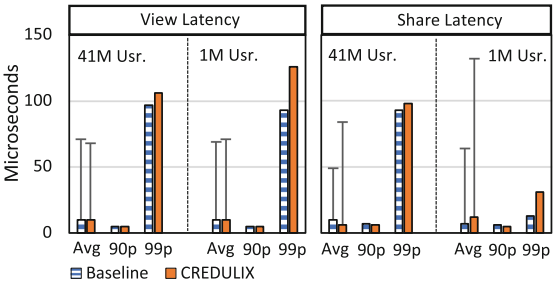


**Fig. 9.** CREDULIX' latency overhead: 99.9% views

The high variation in latency is caused by the intrinsic differences between the users; share operations of a user with more followers need to propagate to more users than posts of users with few or no followers. The high $99^{th}$ percentile latency for both systems results from Twissandra (our Baseline) being implemented in Java, a language (in)famous for its garbage collection breaks. Operations running concurrently with the GC thus experience high latencies. The impact of garbage collection is stronger with CREDULIX than with Baseline, as CREDULIX creates more short-lived objects in memory to orchestrate its background tasks. In addition to the intrinsic differences between users discussed

above, garbage collection also significantly contributes to the high standard deviation observed in all latencies.

## 5 Discussion and Limitations

We believe that Credulix is a good step towards addressing the fake news problem, but we do not claim it to be the ultimate solution. Credulix is one of many possible layers of protection against fake news and can be used independently of other mechanisms. It is the combination of several approaches that can create a strong defense. This section discusses the limitations of Credulix.

**News Propagation.** Credulix does not prevent users from actively pulling any (including fake) news stories directly from their sources. Credulix *identifies* fake news on a social media platform and, if used as we suggest, prevents users from being notified about other users sharing fake news items. Credulix could easily be used to even remove items from users' timelines once identified as fake. As this may raise questions about freedom of speech and censorship, Credulix does not focus on completely removing items.

**Manual Fact-Checking.** Credulix relies on manual fact-checking and thus can only be as good as the fact-checkers. Only users who have been exposed to manually fact-checked items can be leveraged by Credulix. However, fact-checking a small number of popular news items is sufficient to obtain enough users with usable UCRs. Fact-checking a small number of news items is feasible, especially given the recent upsurge of fact-checking initiatives [2–6].

**User Behavior.** Credulix' algorithm is based on the assumption that among those users exposed to fact-checked news items, some share more fake items than others. Analogous assumptions are commonly used in other contexts such as recommender systems. For example, a user-based recommender system would not be useful if all users behaved the same way, i.e. everybody giving the same ratings to the same items.

Our approach shines when the inclination of most users to share fake news does not change too quickly over time. Many systems that are being successfully applied in practice (e.g. reputation systems, or systems based on collaborative filtering) fundamentally rely on this same assumption.

**Malicious Attacks.** The assumption that users do not change their behavior too quickly could, however, potentially be exploited by a malicious adversary. Such an adversary controlling many machine-operated user accounts could deceive the system by breaking this assumption. For example, all accounts controlled by the adversary could be sharing only true reviewed news items for an extended period of time and then suddenly share an un-reviewed fake one (or do the opposite, if the goal is to prevent a truthful news item from being disseminated). Even then, a successful attack would only enable the spread of the news item, without guaranteeing its virality. Moreover, the adversary runs a risk that the fake item will later be fact-checked and thus will appear in their

UCRs, reducing the chances of repeating this attack with another fake item. In fact, the more popular such an item becomes (which is the likely goal of the adversary), the higher the chance of it being fact-checked. The trade-off between false positives and false negatives is expressed by the $p_0$ parameter (see Sect. 2), i.e. the certainty required to flag an item as fake. A high value of $p_0$ might make it easier for the adversary to "smuggle" fake items in the system, but makes it more difficult to prevent true news items from spreading.

**Updating of the Ground Truth.** Like any vaccine, CREDULIX relies on a fraction of fake news to exist in the social network in order to be efficient. If CREDULIX stops the fake news from becoming viral, then the system might lack the ground truth to make future predictions. Hence, there might be periods when fake news can appear again. To avoid such fluctuations, CREDULIX' ground truth should be continuously updated with some of the most current fake news items. CREDULIX' evolution in time, including changes in user behavior as well as updating of the ground truth related to system dynamics, are research directions we are considering for future work.

**Filtering News.** One could argue that removing some news items from users' news feeds might be seen as a limitation, even as a form of censorship. But social media already take that liberty as they display to users only about 10% of the news that they could show. Rather than censorship, CREDULIX should be viewed as an effort to ensure the highest possible quality of the items displayed, considering the credibility of an item to be one of the quality criteria.

## 6 Related Work

CREDULIX shares similarities with *reputation systems* [30], in creating profiles for users (UCRs in CREDULIX) and in assuming that the future behavior of users will be similar to their past behavior. In our approach, however, users are not rated directly by other users. Instead, we compute users' UCRs based on their reaction to what we consider ground truth (fact-checked items).

CREDULIX also resembles *recommender systems* [31] in the sense that it pre-selects items for users to see. Unlike in recommender systems, however, the pre-selection is independent of the requesting user. Our goal is not to provide a personalized selection.

Another approach to detect fake news is to *automatically check content* [17]. Content analysis can also help detect machine-generated fake blogs [26] or social spam using many popular tags for irrelevant content [29]. Another line of research has been motivated by the role of social networks for fake news dissemination in the case of catastrophic events such as hurricanes and earthquakes [21,22].

News item credibility can also be inferred by applying *machine learning* techniques [18], by using a set of reliable content as a training set [33], or by analyzing a set of predetermined features [20]. Other parameters of interest are linguistic quantifiers [34], swear words, pronouns or emoticons [19]. Yet, a malicious agent knowing these specific features could use them to spread fake news.

Facebook received much media attention concerning their politics about fake news. Their first approach was to *assess news sources' reliability* in a centralized way [8]. Recently, Facebook launched *community-based* assessment experiment [9] asking the users to evaluate the reliability of various news sources. The idea is to give more exposure to news sources that are "broadly trusted". Our approach is finer-grained and goes to the level of news *items*. Facebook also used third-party fact checkers to look at articles flagged by users. Very recent work [10] suggests that Facebook started implementing a technique for stopping misinformation which assigns trustworthiness ratings to its users.

*Fact-checking tools* help to annotate documents and to create knowledge bases [12,13]. Curb [24] focuses on the problem of which items to fact-check and when, relying on users to manually flag items. These tools facilitate the fact-checking process that CREDULIX relies on. Like CREDULIX, it leverages the crowd to detect and reduce the spread of fake news and misinformation and assumes a very similar user behavior model. Curb, however, focuses on the problem of which items to fact-check and when, relying on users to manually flag items. Curb only prevents the spreading of items that have been fact-checked. In addition, it assumes fact-checking to happen instantaneously, without taking into account the considerable fact-checking delay.

## 7 Conclusions

We presented CREDULIX, the first content-agnostic system to detect and limit the spread of fake news on social networks with a very small performance overhead. For a more detailed version of the work, including more experiments, explanations and references, see [16].

## References

1. http://money.cnn.com/2016/11/02/media/fake-news-stories/
2. http://www.politifact.com/
3. https://www.snopes.com/
4. https://www.washingtonpost.com/news/fact-checker/
5. https://www.truthorfiction.com/
6. https://fullfact.org/
7. http://www.zettasphere.com/mind-boggling-stats-for-1-second-of-internet-activity/
8. https://www.washingtonpost.com/news/the-switch/wp/2018/01/19/facebook-will-now-ask-its-users-to-rank-news-organizations-they-trust
9. https://techcrunch.com/2018/01/19/facebooks-news-feed-update-trusted-sources
10. https://www.washingtonpost.com/technology/2018/08/21/facebook-is-rating-trustworthiness-its-users-scale-zero-one
11. Cassandra. http://cassandra.apache.org/
12. Documentcloud. https://www.documentcloud.org/
13. Open calais. http://opencalais.com/

14. Twissandra Twitter clone, build on top of cassandra. https://github.com/twissandra/twissandra/
15. Austen-Smith, D., Banks, J.S.: Information aggregation, rationality, and the condorcet jury theorem (1996)
16. Balmau, O., Guerraoui, R., Kermarrec, A.M., Maurer, A., Pavlovic, M., Zwaenepoel, W.: Limiting the spread of fake news on social media platforms by evaluating users' trustworthiness. arXiv preprint arXiv:1808.09922 (2018)
17. Ciampaglia, G.L., Shiralkar, P., Rocha, L.M., Bollen, J., Menczer, F., Flammini, A.: Computational fact checking from knowledge networks. PLoS One **10**, e0128193 (2015)
18. Dewan, P., Kumaraguru, P.: Towards automatic real time identification of malicious posts on facebook. In: PST (2015)
19. Gupta, A., Kumaraguru, P.: Credibility ranking of tweets during high impact events. In: PSOSM (2012)
20. Gupta, A., Kumaraguru, P., Castillo, C., Meier, P.: TweetCred: real-time credibility assessment of content on Twitter. In: Aiello, L.M., McFarland, D. (eds.) SocInfo 2014. LNCS, vol. 8851, pp. 228–243. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13734-6_16
21. Gupta, A., Lamba, H., Kumaraguru, P., Joshi, A.: Faking sandy: characterizing and identifying fake images on twitter during hurricane sandy. In: WWW (2013)
22. Imran, M., Castillo, C., Diaz, F., Vieweg, S.: Processing social media messages in mass emergency: a survey. ACM CSUR (2015)
23. Jenders, M., Kasneci, G., Naumann, F.: Analyzing and predicting viral tweets. In: WWW Companion (2013)
24. Kim, J., Tabibian, B., Oh, A., Schölkopf, B., Gomez Rodriguez, M.: Leveraging the crowd to detect and reduce the spread of fake news and misinformation. In: WSDM (2018)
25. Koch, K.R.: Bayes' theorem. In: Bayesian Inference with Geodetic Applications (1990)
26. Kolari, P., Java, A., Finin, T., Oates, T., Joshi, A.: Detecting spam blogs: a machine learning approach. In: AAAI (2006)
27. Kwak, H., Lee, C., Park, H., Moon, S.: What is Twitter, a social network or a news media? In: WWW (2010)
28. Leskovec, J., Krevl, A.: SNAP datasets: stanford large network dataset collection. http://snap.stanford.edu/data (2014)
29. Markines, B., Cattuto, C., Menczer, F.: Social spam detection. In: AIRWeb (2009)
30. Resnick, P., Kuwabara, K., Zeckhauser, R., Friedman, E.: Reputation systems. Commun. ACM **43**, 45–48 (2000)
31. Resnick, P., Varian, H.R.: Recommender systems. Commun. ACM **40**, 56–59 (1997)
32. Rish, I.: An empirical study of the naïve bayes classifier. In: IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, vol. 3, January 2001
33. Saikaew, K.R., Noyunsan, C.: Features for measuring credibility on facebook information. Int. Sch. Sci. Res. Innov. **9**, 174–177 (2015)
34. Viviani, M., Pasi, G.: A multi-criteria decision making approach for the assessment of information credibility in social media. In: Petrosino, A., Loia, V., Pedrycz, W. (eds.) WILF 2016. LNCS (LNAI), vol. 10147, pp. 197–207. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-52962-2_17
35. Zabell, S.L.: The rule of succession. Erkenntnis **31**, 283–321 (1989)
36. Zhang, H.: The optimality of naive bayes. In: Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, pp. 562–567 (2004)