Received February 1, 2021, accepted February 12, 2021, date of publication February 19, 2021, date of current version March 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3060623

T-CREo: A Twitter Credibility Analysis Framework

YUDITH CARDINALE[1,2], IRVIN DONGO[1,3], GERMÁN ROBAYO[2], DAVID CABEZA[2], ANA AGUILERA[4], AND SERGIO MEDINA[2]

[1]Electrical and Electronics Engineering Department, Universidad Católica San Pablo, Arequipa 04001, Perú
[2]Departamento de Computación y Tecnología de la Información, Universidad Simón Bolívar, Caracas 1080, Venezuela
[3]Univ. Bordeaux, ESTIA Institute of Technology, 64210 Bidart, France
[4]Escuela de Ingeniería Informática, Facultad de Ingeniería, Universidad de Valparaíso, Valparaíso 2340000, Chile

Corresponding author: Ana Aguilera (ana.aguilera@uv.cl)

This research was supported by the FONDO NACIONAL DE DESARROLLO CIENTÍFICO, TECNOLÓGICO Y DE INNOVACIÓN TECNOLÓGICA – FONDECYT as an executing entity of CONSEJO NACIONAL DE CIENCIA, TECNOLOGÍA E INNOVACIÓN TECNOLÓGICA - CONCYTEC under grant agreement No. 01-2019-FONDECYT-BM-INC.INV in the project RUTAS: Robots for Urban Tourism centers, Autonomous and Semantic-based.

ABSTRACT Social media and other platforms on Internet are commonly used to communicate and generate information. In many cases, this information is not validated, which makes it difficult to use and analyze. Although there exist studies focused on information validation, most of them are limited to specific scenarios. Thus, a more general and flexible architecture is needed, that can be adapted to user/developer requirements and be independent of the social media platform. We propose a framework to automatically and in real-time perform credibility analysis of posts on social media, based on three levels of credibility: *Text*, *User*, and *Social*. The general architecture of our framework is composed of a front-end, a light client proposed as a web plug-in for any browser; a back-end that implements the logic of the credibility model; and a third-party services module. We develop a first version of the proposed system, called T-CREo (**T**witter **CRE**dibility analysis framew**o**rk) and evaluate its performance and scalability. In summary, the main contributions of this work are: the general framework design; a credibility model adaptable to various social networks, integrated into the framework; and T-CREo as a proof of concept that demonstrates the framework applicability and allows evaluating its performance for unstructured information sources; results show that T-CREo qualifies as a highly scalable real-time service. The future work includes the improvement of T-CREo implementation, to provide a robust architecture for the development of third-party applications, as well as the extension of the credibility model for considering bots detection, semantic analysis and multimedia analysis.

INDEX TERMS API, credibilty, fake news, information sources, twitter, web scraping.

I. INTRODUCTION

Nowadays, social media generates an immense amount of information, since they are what people mostly use to share and read about a wide variety of topics. In this way, information is shared in free environments that can be used in several contexts, ranging from everyday life, global and local news, to the development of new technologies [1]–[3]. Social media and other platforms on the Internet, which allow users to communicate, share, and generate information without formal references to sources, became popular in the early 1990s, producing such a vast amount of information that fits into the *Big Data* category. However, in many cases, this information is not documented or validated, which makes it tough to use and analyze. Hence, the concept of *credibility*, as the level of belief that is perceived about (how credible it is) a person, object, or process [4], has become essential in various disciplines and from different perspectives, such as information engineering, business administration, communications management, journalism, information retrieval, human-computer interaction [5], [6].

However, existing works are limited to be applicable to analysis of credibility on specific scenarios (e.g., for a specific social platform, for a particular application). These works differ in the characteristics taken into account to calculate credibility (e.g., attributes of the posts or of users who posted them, the text of the posts, user social impact) and in the extraction techniques used to gather the information to feed the credibility models (i.e., web scraping[1] or API). Thus,

The associate editor coordinating the review of this manuscript and approving it for publication was Yassine Maleh.

[1]Web scraping is a technique for extracting information, focusing on the generation of structured data.

a more general and flexible architecture is needed, that can be adapted to user/developer's requirements and be independent of the social media platform.

To overcome these limitations, we propose a framework to automatically and in real-time perform credibility analysis of posts on social media. The framework instantiates a credibility model proposed in our previous work [4], which consists of the credibility analysis of publications on information sources, adaptable to various social networks. The credibility model is based on three aspects: *Text Credibility* (based on text analysis), *User Credibility* (based on attributes about the user's account, such as creation date, verified account), and *Social Credibility* (based on attributes that reflect social impact, such as *followers* and *following*). In this work, we describe the general architecture of the framework and demonstrate its applicability for unstructured information sources, taking as reference Twitter, which is one of the most used among social media networks.

The characteristics of our proposed framework architecture, that make it different from the existing works, are mainly the following:

- It provides two approaches for accessing the information needed for the credibility model: web scraping and social media API; users/developers can configure the system to base the information gathering only with web scraping or combining it with the use of the available API;
- It performs credibility analysis automatically and in real-time;
- It consists of a front-end, which is proposed as a web plug-in to be incorporated on any browser, and a decoupled back-end which executes the credibility analysis;
- It is light-decoupled from external components; as a consequence, it is extensible and flexible; thus, it can be adapted to any social media platform and the credibility model can be extended by replacing or integrating other measures to calculate different credibility levels.

We develop a first version of the proposed system, called T-CREo (**T**witter **CRE**dibility analysis framew**o**rk) as a proof-of-concept. As a Google Chrome Extension, T-CREo performs the credibility analysis of tweets, in real-time. According to the study presented in [7], Twitter statistics indicate that around 500 millions of tweets are published every day. Thus, credibility analysis in such as platform has become a trending topic in the last decades [8]–[11]. There exist many studies proposing Twitter credibility models [4], [8], [11], [12] and more complete studies, which also propose frameworks to perform the credibility analysis automatically and in real-time [13]–[18]. We qualitatively compare our proposal with the state-of-the-art and we show the performance evaluation of T-CREo in various scenarios, with different variables, such as number of requests and number of concurrent clients/connections. Results show that the performance of T-CREo qualifies it as a real-time and highly scalable service.

In summary, the main contributions of this work are: (i) the design of a framework to perform credibility analysis on social networks, automatically and in real-time; (ii) a credibility model adaptable to various social networks, integrated into the framework; and (iii) T-CREo as a proof of concept that demonstrates the framework applicability and allows a comparative evaluation with existing systems and an evaluation of its performance.

The remainder of this work is organized as follows. Section II describes and classifies related studies on the credibility of information from Twitter. Section III summarizes the credibility model used in the framework, which was previously proposed in [4]. Section IV describes the general architecture of our proposed framework and T-CREo, an implementation of the described architecture. In Section V, we analyze the results of a battery of tests, which are made to measure the performance of the proposed architecture and the implementation. We conclude in Section VII.

II. RELATED WORK

Existing works consider the extraction and analysis of different types of information to calculate credibility on social media. Thus, several terms of *credibility* have been proposed [8], [9], [11], [19], [20]. Inspired by these works, we present the following classification of credibility terms in social networks:

- *Text Credibility (Post Credibility):* measures the level of relevance and accuracy of the text, independent of the referenced topic [8] or with respect to a certain topic [11]. It is calculated through text analysis techniques, such as Natural Language Processing (NLP).
- *User Credibility:* calculates the user account credibility based on attributes that describe it. It can be calculated based on, for example, the *account creation date*, if the *account is verified*, *user's age*.
- *Social Credibility:* calculates the credibility of a publication, related or not to a topic, based on the available metadata that describe the social impact of the user account and the *post* itself, with respect to other users. It is calculated based on data such as number of *followers*, number of *following*, *retweets*.
- *Topic-Level Credibility:* measures the level of acceptance of the topic or event referenced in the text. It consists of identifying if the text refers to a specific topic or not, usually through NLP and sentiment analysis techniques.

Together, all these credibility measures attribute a global credibility level of a publication in a social information source; however, usually they are not considered as a whole; some works consider only one aspect or a subset of these measures to calculate credibility. Moreover, most existing works only expose how the credibility model is performed in a **specific context** [4], [8], [11], [12], [20], but they do not propose or describe a **general architecture** to support the process, as it is the focus of our work. This is an important aspect to evaluate the applicability of the proposed approach to be applied in **real-time** applications.

How the measures to calculate credibility are obtained is another issue approached by the existing studies. Some of them base the information extraction on social platform **API** [19], [21], [22], while others use **web scraping techniques** [23]–[26]. Social media API are easy to invoke, but they impose limitations and restrictions of use; while web scraping is much more flexible but also requires more work and should be adapted each time the HTML structure changes. Few recent works have focused on comparing both extraction techniques in Twitter, for credibility analysis [27] or to gather unlimited volume of tweets [24].

More related studies to our work are those focused on proposing a system, an architecture, or a framework to process credibility on social networks. Although, our proposed architecture allows configuring the system according to the social network, since our implementation is for Twitter, we survey in this section such as studies related to Twitter. We compare them in terms of the considered credibility levels, the technique used to extract the information, and their applicability in real-time scenarios.

A. TOOLS FOR CREDIBILITY ANALYSIS ON TWITTER

Some studies have proposed architectures, supported in the Twitter API, to perform Twitter data analytics in real-time [21], [22], [28]–[30] or for pre-processing data for sentiment analysis, as the Service Oriented Architecture (SOA) framework proposed in [31], however they are not in the scope of credibility analysis, as our focus in this work.

Concerning credibility analysis, some studies propose off-line systems able to analyze tweets credibility on user demand [32] or from gathered data [33], [34]; hence, they do not offer real-time credibility analysis. In [32], two measures are proposed to calculate the topic-level credibility; one of them considers the tweet credibility based on the positive and negative opinions of the topic and the other one considers the author expertise. The system consists of four modules and a tweet opinion database: the tweet sender/receiver is an interactive front-end module, which receives the user's input and provides the result to the user; the tweet credibility calculator module utilizes the tweet opinion classifier to identify both the topic and the opinion of a given tweet; then, it performs a majority decision on contrary opinions in the same topic by using the tweet opinion database, which are stored in the tweet collector module and labeled with a topic-opinion label by the tweet opinion classifier module. Hoaxy [33] is a web platform for the tracking of social news sharing. The main goal of this work is to let researchers, journalists, and the general public monitor the production of online misinformation and its related fact checking. The system presents three components: monitors (URL tracker, scrapy spider, RSS parser), a database as repository, and an analysis dashboard. The system collects data from two main sources, news websites and social media, by using web scraping, web syndication, and where available, API of social networking platforms. Afterward, this stored data is analyzed considering two aspects: the temporal relation between the spread of misinformation and fact checking, and the differences in how users share them. Being off-line systems, these platforms are not able to support real-time analysis. An approach to detect Source Of Fake News (SOFNs), by analysing credibility of tweets based on graph Machine Learning, is proposed in [34]. The credibility analysis is based on user features (e.g., created at, name, default profile, default profile image, favourites count, statuses count, description), social graph of users (*followers/following* graph), and topic annotations, whose information is gathered with the Twitter API. Binary Machine Learning classifier models are fed with these features to predict SOFNs.

Some other studies more related to our work, have proposed architectures to perform real-time credibility analysis on Twitter [13]–[18]. In [13], a system for automatically measuring the credibility of Arabic News content published in Twitter is presented. In this system, in addition to considering the text content, characteristics associated with the user account are evaluated, such as its *verified quality* and its Twittergrader.com (i.e., text, user, and social credibility levels). Twittergrader.com measures the power, scope, and authority of a Twitter account based on number of *followers*, *followers* impact, updates, credibility of news, *followers/following* relationship, and commitment. The system architecture is supported by the Twitter API and consists of four main components: text pre-processing, features extraction and computation, credibility calculation, and credibility assignment and ranking.

TweetCred and CredFinder are respectively described in [14], [15], two practical solutions proposed as Google Chrome extensions, which calculate the tweet credibility in real-time, considering the content in the text, attributes of the tweet (publication time, source from which the tweet was posted), and social impact. Both use the Twitter API. TweetCred [14] architecture is composed by a back-end and a front-end; the back-end is responsible of calculating credibility score based on Twitter API to fetch the data about individual tweets and on a Support Vector Machine (SVM) prediction model; the front-end is a local interface embedded in the Google Chrome navigator, from which the tweets IDs are scraped and sent to the back-end and in which credibility scores are shown. The credibility of each tweet is calculated in terms of the text (metadata and content) and social information related to the user (e.g., *followers*, *following*) and related to the tweet (e.g., *retweets, mentions*). CredFinder [15] consists of a front-end in the form of an extension to the Google Chrome browser and a web-based back-end. The former collects tweets in real-time from a Twitter search or a user-timeline page and the latter is based on four components: a reputation-based component, a credibility classifier engine, a user experience component, and a feature-ranking algorithm. Using the Twitter streaming API, tweets and their meta-data (the *time of posting*, the *author name*, number of *followers*, number of *friends*, *hash tags* or *mentions*, etc.) are obtained. All these data are used as input to the credibility score calculation algorithm.

Another web interface framework, implemented as a web plug-in system is proposed in [16]. The aim is to analyze, in real-time, the credibility of tweets regarding to a specific topic. Only the text of each tweet is analyzed to be classified as being entailment, neutral, or contradiction with respect the topic. The system shows a list of news information related to the topic; thus users can decide the veracity of the tweet in question. The Twitter API is used to collect tweets, web scraping is used to get the URLs referenced in tweets, and the Bing news API is used to find articles and retrieve news headlines related to the topic. In [17], a framework for credibility analysis on Twitter data, with disaster situation awareness is proposed. This framework is able to calculate in real-time the topic-level credibility (i.e., emergency situations), by analysing the text, linked URLs, number of *retweets*, and geographic information extracted from both post text and external URLs, which are kept in a database. Thus, an event with a higher credibility score indicates that there are more tweets, more linked URLs, and more *retweets* mentioning this event. Data is collected through Twitter API, to get the information of the tweets and Google Maps Geocoding API to obtain geolocalization information. The text is analyzed, in the event identification module, to extract words that match with keywords that describe a specific event (i.e., a disaster situation topic); then its credibility is calculated by the credibility module.

A more recent system for real-time credibility analysis is described in [18]. This framework considers text and user credibility, aimed to identify fake users and fake news, based on neural network models. Text credibility is measured in terms of *retweets, followers, favorites* (i.e., social impact), and the number of relevant words and the sentiment score (i.e., the text content is analyzed). Users credibility is computed by considering their location, URLs, if the account is verified, the geolocation, the creation date, and the most recent 20 tweets posted by this user. The Twitter API is used to retrieve these data. The system comprises tweets and users monitoring modules, a leaning module, and a credibility module.

B. COMPARATIVE ANALYSIS

All these works described in the previous section, are mainly based on Twitter API and use web scraping only to obtain the tweets IDs. None of them offer both extraction possibilities independently. In contrast, T-CREo framework offers to users the possibility of choosing which extraction technique is the most appropriate to their capabilities (e.g., Twitter API permissions); thus, it can be configured according users requirements.

Table 1 summarizes the comparison of the reviewed works according to the four credibility levels, their applicability for real-time analysis, the extraction technique (API, web scraping), the application scope, and the main characteristics of their implementations. The common characteristics of most revised works is that they do not consider the four levels of credibility and the use of the Twitter API. Related to credibility levels, only the systems proposed in [13], [18], assess credibility in terms of text, user, and social factors, as we do in T-CREo; credibility related to a specific topic is not considered. Although, this aspect grants generality to the systems, we plan to include the topic level in the future, as an option that users can configure. The analysis of the text in T-CREo, is done through filters that detect SPAM, bad words, misspelling. We do not consider yet including Machine Learning algorithms or Sentiment Analysis, as in [16], [17], [32]–[34]. However, such algorithms in the context of specific topics are under consideration in our ongoing research work. T-CREo architecture is flexible enough to easily incorporate them, thereby completing all four aspects of credibility of a tweet.

Concerning the extraction technique, even though all these works use web scraping to get some data (e.g., tweets IDs), they base their data gathering in the Twitter API. Only the system proposed in [33] and our proposal T-CREo, offer web scraping as an independent technique from the API, which makes these systems more flexible and adaptable to users/developers needs and capabilities; however, the system proposed in [33] is applicable only for news verification in specific topics.

Real-time credibility analysis is an important aspect considered in most of these works, including T-CREo. T-CREo separates the system in a light front-end, as a Google Chrome extension, and a back-end, which performs the heavy calculations, similar to TweetCred [14], CredFinder [15], and the system proposed in [16]. A front-end designed as a web plug-in or extension, make it possible to automatically perform the analysis, without the intervention of final users. Additionally, T-CREo front-end is able to recognize, whether the social media is Twitter or Facebook, thus performing the credibility analysis accordingly.

III. CREDIBILITY ANALYSIS MODEL FOR SOCIAL MEDIA

In order to calculate the credibility of a post in a social network, our proposed framework uses the credibility model proposed in a previous work [27]. The credibility measure mainly depends on two components: (i) the post's content: that is a text, which for Twitter is less than 240 characters (at 2020); and (ii) the author: the user that published the post. Features of text and user are extracted to feed the credibility model, which consists of three credibility measures: *Text Credibility, User Credibility*, and *Social Credibility*. Figure 1 shows a general view of such as credibility model. *Text Credibility* is entirely related to the post's text, while *User Credibility* and *Social Credibility* are calculated using users' attributes. Each credibility measure is based on several components that we call *filters*. Hence, the model becomes easy to implement, flexible, and extensible. It does not need advanced data-manipulation, which makes it ideal to use on real-time applications. In the following section, we describe the credibility model in detail.

TABLE 1. Related Work Comparison

Work	Credibility Levels				Real-time	Main Extraction Technique	Scope	Implementation
	Text	User	Social	Topic				
[21], [22], [28], [29], [30]	-	-	-	-	Yes	Twitter API	Data Analytics	Tweets DB data gather and analytic modules
[31]	-	-	-	-	Yes	Twitter API	Sentiment Analysis	SOA framework
[32]	-	User expertise	-	Yes	No	Twitter API	Sentiment Analysis Credibility	Opinions DB, Interactive front-end Credibility and Opinion classifier modules
[33]	URLs	-	-	Yes	No	Twitter API Web scraping	News Verification	Web platform, Tweets DB Monitors and Analysis modules
[34]	-	Creation date Name, Profile Image	*followers/following social graph*	Yes	No	Twitter API	SOFNs detection Credibility	Machine Learning pipeline
[13]	Content	Account verification	*followers, following*	-	Yes	Twitter API	Credibility	Pre-processing, extraction, calculation modules
[14]	Content, source Time of tweet	-	*followers, following, retweets, mentions*	-	Yes	Twitter API	Credibility	Google Chrome extension Back-end for analysis
[15]	Content, source Time of tweet	User reputation	*followers, following, retweets*	-	Yes	Twitter API	Credibility	Google Chrome extension Back-end for analysis
[16]	Content, URLs	-	-	Yes	Yes	Twitter API	Credibility	Web plug-in Back-end for analysis
[17]	Content, URLs	-	*retweets*	Yes	Yes	Twitter API	Credibility (disaster situations)	Tweets/URLs DB Event identif. and credibility modules
[18]	Content, URLs Date of post	Account verification Creation date Location	*followers, retweets, favorites*	-	Yes	Twitter API	Fake users and news identification Credibility	Tweets and users monitoring modules Learning and credibility modules
T-CREo	**Content**	**Account verification Account creation (year)**	**following, followers retweets**	-	**Yes**	**Twitter API Facebook API Web scraping**	**Credibility**	**Google Chrome extension Posts DB, Back-end for analysis**

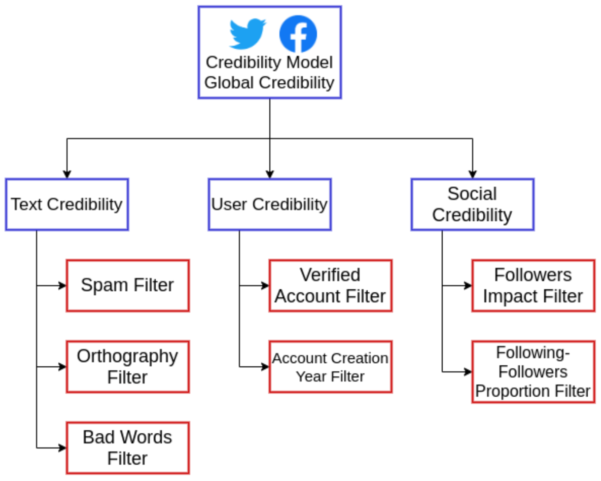


FIGURE 1. Credibility model.

A. TEXT CREDIBILITY

Text credibility analyzes syntactically the content of the post (without checking the author attributes), through SPAM, bad words, and misspelling *filters*, as shown in Def. 1.

Definition 1 (Text Credibility (TextCred)): Given a text of a post, p.text, the Text Credibility is a function, denoted as TextCred(p.text), that returns a measure $\in [0, 100]$*, defined as:*

$$TextCred(p.text) = w_{SPAM} \times isSpam(p.text) + w_{BadWords} \times bad\_words(p.text) + w_{MisspelledWords} \times misspelling(p.text)$$

where:

- *isSpam(p.text) is a SPAM detector that determines the probability* $\in [0, 100]$ *of p.text being a spam;*
- *bad_words(p.text) measures the bad words proportion* $\in [0, 100]$ *against the amount of words in a text;*
- *misspelling(p.text) measures the misspelling errors proportion* $\in [0, 100]$*;*
- w_{SPAM}, $w_{BadWords}$, *and* $w_{MisspelledWords}$ *represent user-defined parameters to indicate the weights that the user gives to each filter, such that:* $w_{SPAM} + w_{BadWords} + w_{MisspelledWords} = 1$.

B. USER CREDIBILITY

User credibility analyzes only the user as a unit of the platform, without being influenced by other users, as it is described in Def. 2.

Definition 2 (User Credibility (UserCred)): Given a set of metadata of a user who published a post, p.user, the User Credibility is a function, denoted as UserCred(p.user), that returns a measure $\in [0, 100]$*, defined as:*

$$UserCred(p.user) = Verif\_Weight(p.user) + Creation\_Weight(p.user)$$

where:

- *Verif_Weight(p.user) is a function that returns 50 if the user is verified and 0 otherwise;*
- *CreationWeight(p.user) measures the time since the user's account was created, with a value between 0 and 50, increasing with the longevity of the account, such that:*

$$CreationWeight(p.user) = \frac{Account\_Age(p.user)}{Max\_Account\_Age(p.user)} \times 50$$

where:

- $Account\_Age(p.user) = CurrentYear - YearJoined(p.user)$
- $Max\_Account\_Age(T) = CurrentYear - SocialPlatform\_Creation\_Year$
- *SocialPlatform_Creation_Year is the year in which the targeted social platform was created (e.g., 2006 for Twitter).*

C. SOCIAL CREDIBILITY

Social credibility is focused on the relations between a user account and the other accounts in the social media platform. It considers the amount of *followers* and *following* (see Def. 3).

Definition 3 (Social Credibility (SocialCred)): Given a set of metadata of a user who published a post, p.user, the Social Credibility is a function, denoted as SocialCred(p.user), that returns a measure $\in [0, 100]$*, defined as:*

$$SocialCred(p.user) = FollowersImpact(p.user) + FFProportion(p.user)$$

where:

- $FollowersImpact(p.user) = \frac{\min(p.user.followers, MAX\_FOLLOWERS)}{MAX\_FOLLOWERS} \times 50$*, measures the impact* $\in [0, 50]$*, on the number of followers;*
- $FFProportion(p.user_{social}) = \frac{p.user.followers}{p.user.followers + p.user.following} \times 50$*, measures the proportion* $\in [0, 50]$*, between number of followers and following of the user.*
- *MAX_FOLLOWERS is a user-defined parameter.*

The *MAX_FOLLOWERS* constant is supplied by the user, for example in [27] it is considered as 2 million. *FFproportion* is self-explanatory, a simple proportion that increases the credibility if the user has more *followers* than *followings*. The purpose of this function is to discredit bots, that tend to have more *following* than *followers*.

D. TOTAL CREDIBILITY LEVEL

The credibility of a post is a weighted sum of the three credibility measures described previously. Def. 4 shows how it is calculated. According to the social network, the respective features for *User Credibility* and *Social Credibility* have to be identified and obtained. Table 2 shows equivalent attributes for Twitter, Facebook, Reddit, Instagram, and LinkedIn.

Definition 4 (Global Credibility Level (GCred)): Given a post, p, the Global Credibility Level is a function, denoted as GCred(p), that returns a measure $\in [0, 100]$*, of its level of credibility, defined as:*

$$GCred(p) = weight_{text} \times TextCred(p.text) + weight_{user} \times UserCred(p.user) + weight_{social} \times SocialCred(p.user)$$

where:

- $weight_{text}$*,* $weigh_{user}$*, and* $weight_{social}$ *are user-defined parameters to indicate the weights that the user gives to Text Credibility, User Credibility, and Social Credibility, respectively, such that:* $weight_{text} + weight_{user} + weight_{social} = 1$*;*
- $TextCred(p.text)$*,* $UserCred(p.user)$*, and* $SocialCred(p.user)$ *represent the credibility measure related to the text, the user, and the social impact of p, respectively.*

E. TIME COMPLEXITY ANALYSIS

Concerning computational complexity, *User Credibility* and *Social Credibility* is performed in constant time, independent of the post's size, that is $O(1)$. However, *Text Credibility* depends on the length of the text. Let n be the amount of words in the text:

- *isSpam* is defined as a function that checks if the text is spam or not, checking every word. Thus the time complexity of this function is $O(n)$.
- *bad_words* needs to iterate over the words of the text and check if each word is a bad word or not. Then, the complexity of *bad_words* is also $O(n)$;
- The analysis of *misspelling* is analogue to the *bad_words*: each word is grammatically verified. Thus, the complexity of *misspelling* is $O(n)$.

Since the three *filters* are sequentially performed, the time complexity of *Text Credibility* is $3 \times O(n)$. Moreover, being $O(1)$ the time complexity of *User Credibility* and *Social Credibility*, the time complexity to calculate the *Credibility Level* of a post is bounded by $O(n)$, with n representing the number of words in the post. Hence, for short texts, as in Twitter, this time complexity does not represent a high consumption of computational resources.

IV. A FRAMEWORK FOR CREDIBILITY ANALYSIS IN SOCIAL MEDIA: OUR PROPOSAL

In this section, we describe the general architecture of our framework to perform real-time credibility analysis on social platforms. Afterward, we present T-CREo, an implementation of our framework for Twitter, as a proof-of-concept.

A. FRAMEWORK ARCHITECTURE

The general architecture of our proposed framework, shown in Figure 2, follows a *Client-Server* pattern. The framework is composed by:

- The front-end, a light client proposed as a web plug-in for any browser, which allows analysing *Global Credibility* in real-time of posts from a social network platform; it also provides the option of analysing a plain text provided by the user, independently of the social platforms, in whose case only the *Text Credibility* is obtained;
- The back-end is the "source of truth" of the framework; it implements the logic of the credibility model and provides mechanisms to calculate each credibility measure;
- Third-party services that groups the services that the back-end consumes directly and are not of our intellectual property; they are our *data-sources* to calculate the

TABLE 2. **Available Attributes for Social Networks**

	Attribute	Twitter	Facebook	Reddit	Instagram	LinkedIn
Text credibility	Text	Text for analysis	Text for analysis	Text for analysis	Text for analysis	Text for analysis
User credibility	VerifWeight	Account verification	Account verification	-	Account verification	Premium user
	Year joined	Creation day	Creation day	Creation day	-	-
Social credibility	Social user followers	Followers	Friends	Followers	Followers	Connections
	Social user following	Followings	Mutual friends	Followings	Followings	Connections

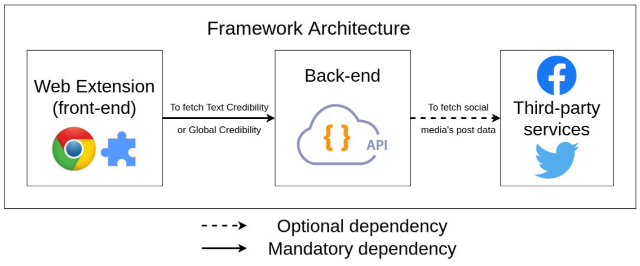


FIGURE 2. **General architecture of the proposed framework.**

credibility of external *entities* in the credibility model; on this module, we have applications like Twitter and Facebook, which provides all the information related to posts and users of the platform.

The front-end, a web extension (*client*), sends requests for *Global Credibility* calculation along with some parameters to the back-end. In the case of users providing a plain text, it requests *Text Credibility* analysis. The back-end (*server*) receives the request and returns a response, based on its contents, to the front-end. The back-end is also a client of the third-party services web module, which is not under our control. Arrows in Figure 2 define the dependency relationship among the components from a high-level view.

Relationship between the front-end and back-end is mandatory, while the connection to a third-part service is optional and is only needed when the respective API is used. The optional dependency implies that the application can still be usable without access to social network API, but our proposal *needs* the back-end to be able to connect to that. Note that third-party services module is not exclusively limited to Twitter or Facebook.

The credibility model can be adapted to any other social network. Thus, it is possible to add even more information-sources (usually social media) accessible through the back-end. In the following, we describe deeply the front-end and the back-end of the framework.

1) FRONT-END

In our proposed architecture, the front-end is a web plug-in or a browser extension to avoid user-intervention in the process of obtaining the *Global Credibility* of posts in a social network platform. In this way, it is able to do web scraping and to make HTTP requests. The front-end is logic-less – i.e., it does not perform any calculation related to the credibility model, which is delegated to the back-end. It allows users to configure and set all user-defined parameters to adjust the results to their needs. It is able of recognizing whether the social platform is Facebook or Twitter and permits to configure the data extraction mechanism to gather the information to be sent to the back-end (i.e., web scraping of all information or scraping of the post IDs combined with the social media API). When the web scraping option is selected, the front-end sends the request of performing *Global Credibility* analysis, along with all parameters needed for the credibility model (e.g., number of *followers*, date of creation of the account). In this case, such parameters are obtained with web scraping at the front-end. In contrast, if the API option is selected, the *Global Credibility* request is accompanied only with the post IDs. In addition, it provides an interactive option for users to demand *Text Credibility* analysis of provided plain texts, independently of the social platforms.

Implementation of the front-end as web, mobile, or desktop applications are not considered, because they do not allow performing web scraping, which is a requirement in our proposal. However, this type of *client* can interact with the back-end, through its API (REST or SOAP), by manually providing the required parameters (e.g., post IDs, user IDs) to gather information and get successful responses.

2) BACK-END

The reason why the back-end is separated from the front-end is that, this way, any external agent from the architecture can access it. It also makes the system more extensible, since it does not have to replicate the logic of the model, only request it. If at any moment there is the need to add another module, it only has to know how to access and use the back-end resources, without knowing what happens behind, making everything in a more declarative way. It offers *endpoints* to access the mechanisms to calculate all credibility measures, based on REST API or SOAP API (although we recommend the prior because of its flexibility, wide use, and support).

The architecture of the back-end, as shown in Figure 3, is based on a *layered* pattern:

- The first layer corresponds to the *Controllers*, that contains the REST or SOAP API *endpoints* that the back-end exposes. Each *endpoint* is responsible for making data serialization and de-serialization, validation, and make the call to the *domain layer*. There should be at least one *endpoint* to calculate *Text Credibility* for plain texts provided by users and one to calculate *Global Credibility* of posts of a social network platform. *Global Credibility* requests can be performed through *endpoints* with any of the following purposes:

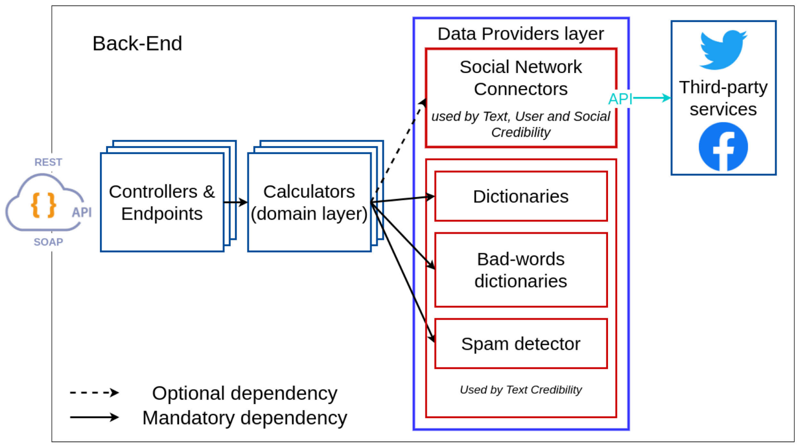


FIGURE 3. **General architecture of the back-end.**

– Calculate *Global Credibility* by gathering all credibility model parameters from the social platform API.
– Calculate *Global Credibility* without calling the social platform API – i.e., all needed arguments are received from the front-end along with the requests; the front-end collects all parameters based on web scraping.

The framework offers this choice to allow the development of implementations even when access to the social platform API is not possible. Notice that the proposal does not require to implement *endpoints* to calculate *Social Credibility* and *User Credibility* separately, since the front-end requests *Global Credibility*, which in turn internally invokes the corresponding *Calculators* for all credibility measures, at the *domain layer*.

- *Calculators* at the *domain layer* implement the credibility model. There is a *Calculator* for each credibility aspect, that invokes the *Data Providers* layer, if it is necessary.
- Each module of the *Data Providers* layer corresponds to a part of the credibility calculus. It is not mandatory that the back-end implements each module, since the functionality can be provided by third-party libraries. For example, in the case of web scraping option, for *Text Credibility*, *IsSpam*, *bad_words*, and *misspelling* filters are implemented in this layer; however, for *User Credibility* and *Social Credibility*, there is nothing needed at this layer. In contrast, in the case of API option, besides the *Text Credibility* filters, *Social Network Connectors* should be implemented at this layer to gather the needed parameters from the third-party services.
  The *Social Network Connector* acts as a bridge between our REST or SOAP API and the social platform API, querying the necessary information to calculate *User* and *Social Credibility* that is needed for *Global Credibility*. Notice the *optional* dependency between the *Calculators* layer and the *Social Network Connector* module makes the framework works without having access to social platform API.
  The rest of the modules of the *Data Providers* layer, namely *Dictionaries*, *Bad words dictionaries*, and *Spam detector*, are used for the *misspelling*, *bad_words*, and *IsSpam* filters of the *Text Credibility*, respectively.

The following section describes an implementation of our proposed framework for Twitter.

B. T-CREo: AN IMPLEMENTATION OF THE CREDIBILITY ANALYSIS FRAMEWORK

In this section, we describe T-CREo (**T**witter **CRE**dibility Framew**o**rk),[2] an implementation of the proposed framework architecture to perform analysis of credibility on Twitter. We describe each component as follows.

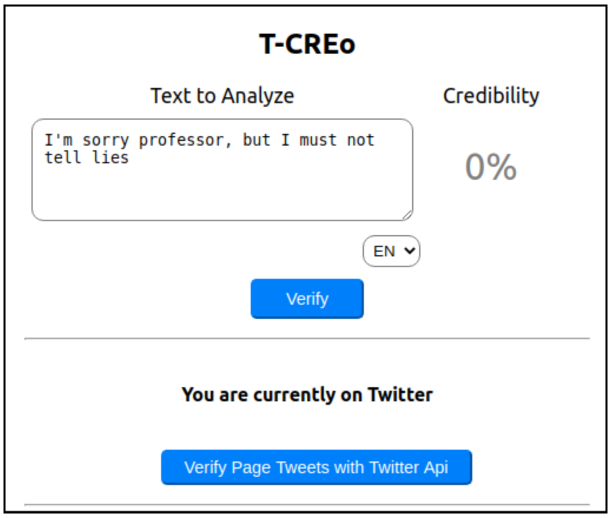
1) FRONT-END

The front-end of T-CREo is an extension for Google Chrome. Accordingly to our proposal, the front-end allows users to:

- Analyze *Text Credibility* of a provided text.
- Configure the parameters required for the credibility model.
- Select the data extraction technique – i.e., users decide if they want to base the data gathering on Twitter API or web scraping.

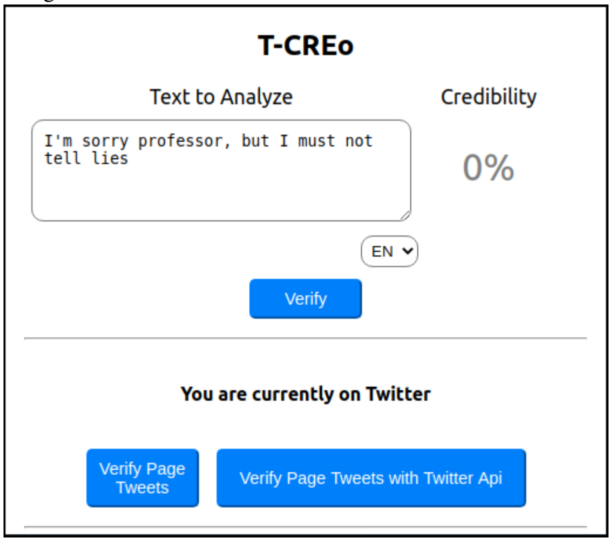
The web extension detects the website (`twitter.com` or `facebook.com`) on the current browser tab and updates its user interface accordingly. In this version of T-CREo, the `facebook.com` option is off. In the case of `twitter.com`, if the page is the *home timeline*, it shows *only* the option to verify tweets using the Twitter API (see Figure 4a); while if it is on an *user's profile timeline*, the option to verify tweets by using web scraping appears next to the option to use Twitter API (see Figure 4b). The reason of only showing the Twitter API option at the *home timeline* is because there is no way to scrap the user and account data

[2]Github repository: https://github.com/t-creo


T-CREo
Text to Analyze
Credibility
0%
EN
Verify
You are currently on Twitter
Verify Page Tweets with Twitter Api

I'm sorry professor, but I must not tell lies

(a) Home timeline with only the option to calculate credibility using Twitter API


T-CREo
Text to Analyze
Credibility
0%
EN
Verify
You are currently on Twitter
Verify Page Tweets
Verify Page Tweets with Twitter Api

I'm sorry professor, but I must not tell lies

(b) User's timeline with the options to calculate credibility using Twitter API or only web scraping

FIGURE 4. **Front-end on** `twitter.com`**.**

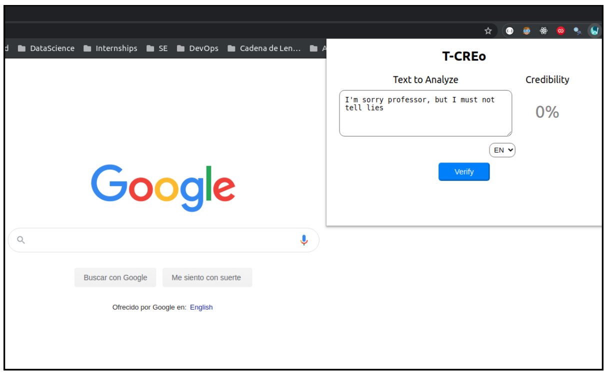
from each tweet's author, required to calculate the tweets' credibility, only the *tweet IDs* are accessible via web scraping; thus, the rest of the attributes must be gathered via the Twitter API. On any Twitter user's profile, for full web scraping data gathering, the front-end assumes that all tweets in that timeline are from the user authorship, including retweets (retweets can be considered as a *copy* of another tweet). In that way, we can collect all user related information of the tweets from the header of the user's profile web page and the specific parameters of each tweet (i.e., text and language) from each single tweet in the feed.

After selecting the extraction technique, either web scraping or Twitter API, the front-end analyzes the current timeline, extracting the necessary information, sending them to the back-end, and finally presenting the results over the timeline. Table 3 describes how each argument is scraped from the HTML using Javascript, when web scraping is selected as the extraction data technique. If the user chooses to verify tweets using Twitter API, only the *tweet IDs* are scraped. Section IV-B3 describes in detail how the model attributes are extracted via Twitter API.

TABLE 3. **Data Extraction Attributes via Web Scraping**

Attributes	Scraper
Text	Array.from(document.querySelectorAll (’div[data-testid="tweet"]’))[i]. children[1].children[1].children[0].innerText
Verified	document.querySelector (’svg[aria-label="Verified account"]’)
Account joined year	let x = document.querySelector (’div[data-testid= "UserProfileHeader_Items"]’).children[i] if x.textContent.match(/^(Joined)/) { x.textContent.split(’ ’)[2]}
Followings	const followingPath = window.location. pathname + ’/following’ document.querySelector(‘a[href= "${followingPath}"]‘).getAttribute(’title’)
Followers	const followersPath = window.location. pathname + ’/followers’ document.querySelector(‘a[href= "${followersPath}"]‘).getAttribute(’title’)

Tweet IDs	const times = document
	.querySelectorAll(’div[data-testid="tweet"]
	time’) const tweetIds = []
	for (let i = 0; i < times.length; i++) {
	const x = times[i].parentElement.
	getAttribute(’href’)
	if (x) { tweetIds.push(x.split(’/’)[3]) }
	}

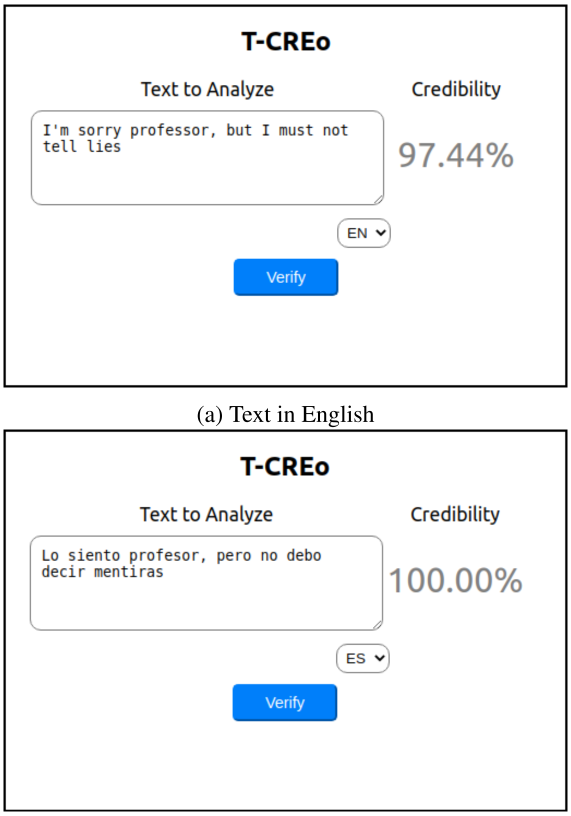


I'm sorry professor, but I must not tell lies

FIGURE 5. **T-CREo front-end as a Google Chrome Extension when opened in any website that is not Twitter.**

Although the front-end starts to automatically doing the credibility analysis of tweets in the timeline of the current account, it offers the possibility of analysing a provided text, based on the *IsSpam*, *bad_words*, and *misspelling* filters, even if the user is not in the Twitter website.

Figure 5 shows how the extension looks when the user is on any website that is not Twitter or Facebook. On the text analysis window, the user can enter the text, select the



T-CREo

Text to Analyze | Credibility

I'm sorry professor, but I must not tell lies

97.44%

EN

Verify

(a) Text in English

T-CREo

Text to Analyze | Credibility

Lo siento profesor, pero no debo decir mentiras

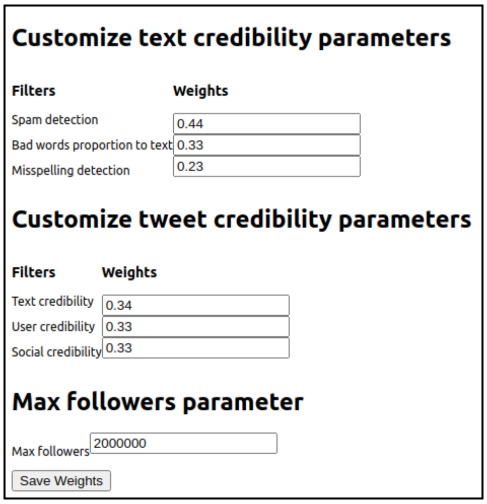
100.00%

ES

Verify

(b) Text in Spanish

FIGURE 6. **Response after clicking *Verify* button.**



Customize text credibility parameters

Filters	Weights
Spam detection	0.44
Bad words proportion to text	0.33
Misspelling detection	0.23

Customize tweet credibility parameters

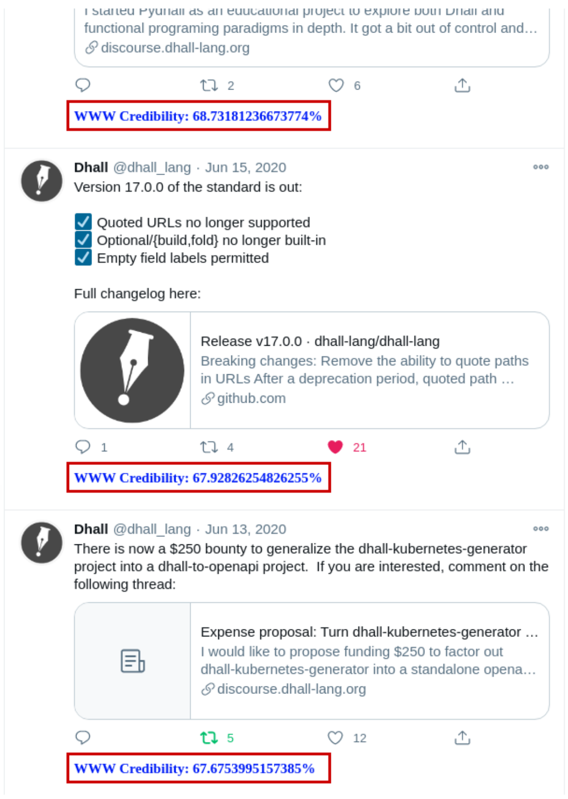
Filters	Weights
Text credibility	0.34
User credibility	0.33
Social credibility	0.33

Max followers parameter

Max followers 2000000

Save Weights

FIGURE 7. **Parameters configuration on T-CREo front-end.**



I started Pyunait as an educational project to explore both Dhall and functional programing paradigms in depth. It got a bit out of control and…
discourse.dhall-lang.org

2 6

WWW Credibility: 68.73181236673774%

Dhall @dhall_lang · Jun 15, 2020
Version 17.0.0 of the standard is out:

✓ Quoted URLs no longer supported
✓ Optional/{build,fold} no longer built-in
✓ Empty field labels permitted

Full changelog here:

Release v17.0.0 · dhall-lang/dhall-lang
Breaking changes: Remove the ability to quote paths in URLs After a deprecation period, quoted path …
github.com

1 4 21

WWW Credibility: 67.92826254826255%

Dhall @dhall_lang · Jun 13, 2020
There is now a $250 bounty to generalize the dhall-kubernetes-generator project into a dhall-to-openapi project. If you are interested, comment on the following thread:

Expense proposal: Turn dhall-kubernetes-generator …
I would like to propose funding $250 to factor out dhall-kubernetes-generator into a standalone opena…
discourse.dhall-lang.org

5 12

WWW Credibility: 67.6753995157385%

FIGURE 8. **Result of pressing any button under an account's timeline.**

language of the text (i.e., English, Spanish, or French), and ask for credibility analysis. The *Verify* button makes the request to the back-end to perform *Text Credibility* over the text that the user has supplied. After clicking the *Verify* button, the response looks like on Figure 6a for a text in English and on Figure 6b for a text in Spanish.

Through the T-CREo front-end, the user can see and edit the parameters for the credibility model (see Figure 7). It verifies that the sum of the weights on each section equals 1. Users can access this configuration window by right-clicking the extension icon in the toolbar, then selecting *Options* or by navigating to the extension management page at `chrome://extensions`, locating the desired extension, clicking *Details*, then selecting the *Options* link.

Figure 8 shows a capture from the timeline of *@dhall_lang*[3] account, that on that date (June 26, 2020) it is *not verified*, has 1338 *followers*, and 0 *following*. The user-defined parameters for the credibility model for this case are presented in Table 4.

2) BACK-END

T-CREo back-end is a REST API developed using Express.js[4] that implements the proposed architecture for credibility calculation. It exposes several *endpoints*, but the most important are:

- `/plain-text`: it receives the necessary parameters to calculate the *Text Credibility*, which are:
  - The weight of each filter, three in total: w_{SPAM}, $w_{BadWords}$, and $w_{MisspelledWords}$.
  - The text to analyze, *t.text* or just a provided *text*.
  - The language of the text. At this moment, it supports English, Spanish, and French. It is supplied via the

[3]https://twitter.com/dhall_lang
[4]https://expressjs.com/

TABLE 4. Parameters Used to Calculate Credibility on *@dhall_lang* Account's Timeline

Parameter	Value
$MAX\_FOLLOWERS$	1500
$weight_{spam}$	0.40
$weight_{misspelling}$	0.30
$weight_{bad\_words}$	0.30
$weight_{text}$	0.40
$weight_{social}$	0.30
$weight_{user}$	0.30

lang query parameter and the possible values are `en`, `es`, and `fr`. The text language is necessary to correctly choose orthography and bad words dictionaries. Currently, we do not have access to a language detection API or library.

This *endpoint* corresponds to the leftmost branch on Figure 1.

- `/twitter/tweets`: it receives the parameters to calculate the credibility of a tweet. It uses the same parameters from the `/plain-text` resource and the following ones:
  - The weight of each filter, three in total: $weight_{social}$, $weight_{user}$, and $weight_{text}$.
  - The *tweet ID* to analyze.
- `/twitter/scraped`: this *endpoint* behaves similar to the `/twitter/tweets` *endpoint*. The difference is that, unlike the latter, this *endpoint* does not request any information from the Twitter API but requires passing the attributes explicitly via query parameters. This is useful to test the whole model since the Twitter API has certain usage restrictions[5] per *endpoint* that does not allow making a lot of requests in a 15 minute window, and for consumers there may be the case that there is no way to obtain the *tweet ID* but all of the other information to calculate the *Global Credibility*.
  This *endpoint* receives the weight of each filter, and the following attributes:
  - If the tweet's author is verified;
  - Account creation year;
  - Amount of *followers* of the author;
  - Amount of accounts this author follows (*followings*).

Both `/twitter/tweets` and `/twitter/scraped` *endpoints* use the same method to calculate the *Global Credibility* of tweets. They both correspond to the whole credibility model shown in Figure 1. These *endpoints* correspond to the *Controllers* layer from Figure 3. The logic behind each *endpoint* is on the *Calculators* layer, where every filter from the credibility model proposed on [35] is implemented as a Javascript function. Each filter function just receives the parameters without worrying about *how* those were gathered. Thus, it is possible to reuse the same filter on each *endpoint* without duplicating code: what really changes between each *endpoint* is if the parameters were gathered through the Twitter API or by web scraping. These filters also interact with the *Data Provider* layer, specifically to invoke the *Text Credibility* filters.

Since our API is written in Node.js v1.16.0,[6] we have access to the whole NPM package ecosystem[7]. The dictionaries used for the *misspelling* filter of *Text Credibility* are provided by Wooorm/dictionaries,[8] a GitHub repository containing dictionaries for several languages. To detect if a word is a *bad word*, we use the `washyourmouthwithsoap` library in NPM package, that is in essence a database of bad words with a method to lookup a word on a specific language. For SPAM detection, we use a fork of the `simple-spam-filter` package in NPM, due to the authors of the package stopped to maintain it on 2019. Our fork[9] uses the original logic, replacing the dictionary libraries with the ones from Wooorm/dictionaries. To connect with Twitter API, we use `twit`,[10] which provides a Node.js API client.

More features can be added to the credibility model by enhancing our back-end. One of those enhancements is for the *Social Credibility* filter by gathering historical tweets from an user. This can be done by running a CronJob (a Unix tool), that gathers the latest tweets from a set of users. This set could be manually defined or automatically updated every time a tweet from an unknown author is analyzed by the `/twitter/tweets` *endpoint*. Having these historical data, we can tune the model to give a more accurate score. Other tweet's attributes can be also taken into account, such as the amount of *retweets* and *likes* in a tweet, that can influence its credibility. Another improvement more related to the *Text Credibility* is to perform sentiment or context analysis on the tweets and study how that affects credibility.

3) THIRD-PARTY SERVICES

At this moment, T-CREo only uses Twitter as third-party web service. The Twitter API access is done on its website.[11] From this API, T-CREo uses the following two resources:

- `GET users/show/:user_id`, that receives a *user_id* and returns the related information to that user. From this resource we use the following fields:
  - `verified`, that is a *boolean* indicating if the user is verified or not;
  - `created_at`, that is the date when the user account was created;

[5]https://developer.twitter.com/en/docs/twitter-api/v1/tweets/post-and-engage/api-reference/get-statuses-show-id

[6]https://nodejs.org/en/

[7]https://www.npmjs.com/

[8]https://github.com/wooorm/dictionaries

[9]https://github.com/t-creo/back-end/blob/54454ff3c927dfb932083c50d97732e1a3676519/src/calculator/spam-filter.ts

[10]https://github.com/ttezel/twit

[11]https://developer.twitter.com/content/developer-twitter/en

  - `followers_count`, that is the amount of *followers* of the user;
  - `friends_count`, that is the amount of accounts that the user follows.
- `GET statuses/show/:id`, that receives a *tweet ID* and returns all its information, including the one of the author. From this resource we use the following fields:
  - `full_text`, that contains the tweet's text;
  - `lang`, which is the tweet's language. This is an important parameter to perform a correct calculus of *Text Credibility*;
  - All of the aforementioned fields from the `GET users/show/:user_id` section.

Each *endpoint* has an usage limit, but it is not a limitation at this moment, since the application is currently on a testing-phase. We are planning to create a tweet database to keep more recent tweets and perform more experiments.

In the next section, we present the experiments to evaluate the performance of T-CREo.

V. EXPERIMENTAL EVALUATION

The experimental evaluation aims to present a quantitative analysis of T-CREo's performance over variables, such as the number of requests and the number of concurrent clients. To be specific, we perform tests that involve simulated users making requests to verify whether T-CREo framework can support the anticipated load and to measure its performance and scalability. The focus is to show how efficiently T-CREo behaves under the pressure of a certain number of requests and concurrency level. The script to perform the evaluation is available here.[12]

A. ENVIRONMENT AND TESTS SETUP

For the deployment of T-CREo framework, we used DigitalOcean cloud services.[13] Servers on DigitalOcean are called *droplets*. We hosted T-CREo back-end in four DigitalOcean droplets located in San Francisco, CA (SF), and New York, NY (NY), as Datacenter regions. We have implemented an automated deployment of the framework within docker containers[14] (deploy.sh and TravisCI) by picking the Docker server version 19.03.12, with Ubuntu 20.04. The droplets are referred to as *High* (i.e., *SF High* and *NY High*) for dedicated CPUs, while those denoted as *Low* (i.e., *SF Low* and *NY Low*) have shared CPUs. *High* droplets have the following characteristics:

- Price: $320/month
- RAM/CPU: 32GB / 16 CPUs
- Disk: 200 GB SSD disk
- Network: 7 TB outbound data transfer (inbound bandwidth to droplets is always free).

On the other hand, the *Low* droplets features are:

- Price: $5/month
- RAM/CPU: 1GB / 1 CPU
- Disk: 25 GB SSD disk
- Network: 1 TB outbound data transfer (inbound bandwidth to droplets is always free).

Under normal conditions, the web extension (i.e., T-CREo's front-end) is responsible for scraping the tweet data and making requests to the back-end to obtain the credibility score. This sequence of operations include local in-memory read operations (at the front-end), Internet access to send/receive the front-end's request/response to/from the back-end, and the back-end operations to calculate the credibility score. Since local read operations are performed at the front-end in despicable times (< 0.1ms) and to facilitate the tests, we developed a shell script that simulates the front-end tasks, called script-client. Thus, the script-client can be executed at the same droplet where the back-end is executed to avoid Internet access impact in the evaluation of the back-end performance. Non-local tests involve Internet latency which prevents analyzing the behavior of the back-end in a controlled environment. Nonetheless, we also execute the script-client in a different machine to measure the Internet impact.

We use Apache Benchmark,[15] a tool for evaluating the performance and behaviour of Apache HTTP servers, to execute the tests. In particular, it shows how many requests per second an Apache installation can handle. In the script-client, we embed the data of the tweets that we randomly select. Thus, Apache Benchmark makes the requests to the back-end and captures the performance measures, with which we perform the analysis, supported on the range of multiple provided options. For our evaluation, we used concurrency (-c) and requests (-n):

- *Concurrency (-c):* represents the number of multiple requests to perform at a time. By default, Apache Benchmark executes one request at a time.
- *Requests (-n):* specifies the number of requests to perform for the benchmarking session. The default is to just perform a single request which usually leads to non-representative benchmarking results.

In our experiments, we test with 5, 10, 20, and 30 levels of concurrency (or number of simultaneous connections/clients), each one with 50, 100, 200, 500, 1000, and 2000 requests (i.e., 24 different experiments). The number of requests is evenly divided between the simultaneous clients. Since the back-end is implemented as a JavaScript runtime, which is single-threaded, the concurrency level represents the simultaneous connections that the back-end has to manage; it does not represent the number of attended simultaneous requests. These 24 experiments were performed in six different scenarios:

- *Scenario a:* Two local scenarios in SF droplets: the script-client and the back-end are executed at the same Low and High droplet, denoted as *Local SF Low* and *Local SF High*.

[12]https://github.com/t-creo/back-end/blob/develop/scripts/script-evaluation.sh
[13]https://www.digitalocean.com/
[14]https://github.com/t-creo/back-end/blob/develop/deploy.sh

[15]https://httpd.apache.org/docs/2.4/programs/ab.html

- *Scenario b:* Two remote scenarios with machines in SF: the script-client is executed in a machine different from the back-end, resulting as *SF Low-SF Low* and *SF High-SF High*.
- *Scenario c:* Two remote scenarios with the script-client executed in a *SF Low* machine and the back-end in NY *Low* and *High* machines, having *SF Low - NY Low* and *SF Low -NY High*. In this scenario, we execute the script-client at the *Low* machine configuration, since users can access the service from mobile devices or desktop PCs, which are not necessary powerful machines.

In total, we present in Section V-C results for $24 \times 6 = 144$ experiments.

Although T-CREo framework offers two methods to extract the information and to calculate the credibility (i.e., web scraping and Twitter API), we did not test the *endpoint* that uses the Twitter API, as it differs from web scraping only in the technique to fetch the data. Furthermore, calls to the Twitter API are limited and incur in network latency, which is not part of the scope of our discussion. In addition to having two methods to obtain a credibility score, T-CREo has *endpoints* to calculate the specific scores of the credibility formula, i.e., *Text, User*, and *Social* scores. In the experimental evaluation, we only test the `/calculate/tweets/scraped` *endpoint*, since it encompasses all the operations of those *endpoints*.

The input parameters remain constant throughout the tests, to minimize the complexity and to obtain a more accurate comparison of the results across the tests. The input parameters for the `/calculate/tweets/scraped` *endpoint* are exhibited in Table 5. The *User Credibility* and *Social Credibility* parameters (e.g., *followers*, *following*, *verified*) are attributes of a randomly selected user at the time of the study, and all weight parameters (i.e., w_{SPAM}, $w_{BadWords}$, and $w_{MisspelledWords}$) were also chosen randomly.

B. TEST METRICS

Apache Benchmark has a multiple range of options to perform the request and return various measures in its output. Since the focus of our tests is performance and scalability, we use the following metrics:

- *Time Per Request:* This value is the average time spent per request.
- *Number of Requests Per Second:* This metric is the result of dividing the number of requests by the total time taken to resolve all of them.

C. RESULTS

Table 6 shows the average time per request for the 144 experiments. In **scenario a**, where the network access is not considered (i.e., the script-client and the back-end are executed in the same droplet), the capacity of the server (*Low* or *High*) has not any impact in this metric. Results are quite similar in all experiments in this scenario; the average is **1.87ms** and **1.57ms** for *SF Low* and *SF High*, respectively. The difference between MIN and MAX values (≤2ms) observed in this scenario, for both cases, evidence that the number of requests and connections (i.e., Concurrency) do not affect the behavior of the back-end; it keeps stable, independently of these factors. Then, under this scenario, the back-end is highly scalable: number of requests and level of concurrency do not impact the time per request. Although these local tests do not represent an environment fully adapted to a real-life use case, because this environment means that the user is deploying the server on their local machine, these tests allow analyzing the performance of the back-end ignoring factors, such as latency, bandwidth, geographic region of the user.

Results in **scenario b** and **scenario c**, show that the effect of Internet access is also impacted by the capacity of the server and the proximity of locations between client and server. In **scenario b**, with *High* capacity and closeness location (i.e., *SF High - SF High*), the back-end is able to attend each request as in the local **scenario a**, with an average of **1.43ms** per request, and it remains stable independently of Requests and Concurrency (the difference between the **MIN** and **MAX** values is very low: **MAX-MIN = 1.21ms**). This is the best result obtained for the three scenarios, representing an average up to 23% of improvement with respect to the local **scenario a**; this is because in **scenario a**, the back-end is sharing resources with the script-client. Meanwhile, with *Low* capacity, even though client and server are close (i.e., *SF Low - SF Low*), the average time per request increases around 53% and 64%. Thus, under these conditions, scalability is bounded by the capacity of the servers.

In **scenario c**, the fluctuations of the back-end increases, reaching a difference up to 73.59ms between the MIN and MAX observed values (for *SF Low - NY Low*). As in

TABLE 5. Input Parameters for the /Calculate/Tweets/Scraped *Endpoint*

Parameter	Value
followers	40418926
following	103
Language	English
MAX_FOLLOWERS	2000000
tweet.text	Nothing can escape gravity, not even black holes (and they really tried!)
verified	true
YearJoined	2009
$w_{BadWords}$	0.33
$w_{MisspelledWords}$	0.23
w_{SPAM}	0.44
$weight_{text}$	0.34
$weight_{user}$	0.33
$weight_{social}$	0.33

previous scenarios, the number of requests does not affect the behaviour of the back-end, however the level of concurrency generates different results: as concurrency increases, the time per request decreases. Although the level of concurrency does not represent simultaneous attended requests, but simultaneous connections, the improvement is due to the fact that the greater the number of connections, the more requirements are going through the network, achieving overlapping of computation with communication. Thus, the server waits less for the requirements to arrive; they are already on its machine, when it finishes one request. Hence, the back-end remains scalable in these conditions.

In general, these results show that the average time per request increases as the remoteness increases and the server capacity is lower. Nevertheless, the level of concurrency allows the back-end remains scalable; in fact, as the level of concurrency increases the time per request decreases. This result can be better appreciated in Figure 9. Note that the time per request in scenarios in which network access has not impact (i.e., *Local SF Low*, *Local SF High*, and *SF High - SF High*) does not highly vary across Request and Concurrency, while time per request for scenarios in which Internet access has impact (i.e., *SF Low - SF Low*, *SF Low NY Low*, and *SF Low - NY High*) does not highly vary as number of request increases and, instead decreases as the level of concurrency increases.

Table 7 shows the results for the same 144 experiments, but measuring the number of requests per second. As expected, best results are obtained in scenarios where the bests time per request were obtained: *Local SF Low*, with **569.55 requests/s**; *Local SF High*, with **664 request/s**; and *SF High - SF High*, with **716 request/s**. The worst result is for *SF Low - NY Low*, with **52 request/s**, because of the network access; this impact is reduced by the number of connections, as we explained before (see Figure 10).

This battery of experiments demonstrates that the back-end is scalable, even though the CPU capacity is not fully used: *Low* capacity droplets reach, in average, 40% of use of the CPU, while the *High* capacity ones, barely consume 0.5% of the CPU. Better management of resources, such as multi-threading to actually serve several requests simultaneously, should improve even more these results.

Concerning the real-time capacity of T-CREo, we consider the definition provided in the Realtime API Hub site[16]: *real-time refers to a synchronous and bi-directional communication channel between endpoints at a speed of less than 100ms*. Although 100ms is an arbitrary number, experiments conducted with 81.000.000 people determined that the median reaction time for humans is 273ms, whereas average reaction time is 284ms. Anything below this value is considered to deliver a satisfactory user experience.[17] Since the highest average time per request obtained in our experiments is **24.67ms** and the highest maximum obtained value is **83.88ms** (see rows **AVG** and **MAX** in Table 6, respectively), we can conclude that T-CREo performs in real-time.

VI. DISCUSSION

T-CREo implementation demonstrates the feasibility of a scalable system for real-time credibility analysis in social networks. This experience also gives the opportunity of extracting its current limitations and some lessons learnt.

A. IMPROVE THE PERFORMANCE

As outlined in Section IV-B2, the back-end is developed in Express, a Node.js framework implemented as *a JavaScript runtime built on Chrome's V8 JavaScript engine*. Javascript is a single-threaded language and it has not a native way of creating threads to *parallelize* the work. A possible solution available in Node.js to overcome this limitation is the usage of asynchronous tasks, which are frequently used to parallelize I/O operations, such as reading files and network calls. All *endpoints* in the current version of T-CREo are implemented as synchronous tasks, since they do not perform network requests or file readings. Making a synchronous task behaves as an asynchronous task is counterproductive and not recommended. Accordingly, T-CREo's *endpoints* are able to handle only one request at a time. Concurrency level in the experiments represents the simultaneous connections that the back-end has to manage; it does not represent the number of attended simultaneous requests.

This limitation affects the framework performance, as shown in Section V: the server with the lowest resources consumes $\sim$ 40% of the CPU, while on the one that has the largest resources, it barely consume $\sim$ 0.5%. It is obvious that we should take actions to make a better usage of the resources of back-end servers. Since, asynchronous tasks are not an appropriate alternative to implement concurrency, other possibilities should be tried, such as:

- Run several replicas (or instances) of the server and let a scheduler web server (such as *Nginx*[18]) attends the requests in a round-robin fashion or in any other available scheduling strategy. This way, it can mimic multi-threads and parallelism by handling several requests at the same time.
- Use a new experimental feature of Node.js, named *Worker Threads*,[19] to implement multi-threaded applications in a single node process. The downside is that, for our current Node.js version, it is a non-stable feature. Nonetheless, it is stable for the latest long-term support version of Node.js v14.15.0, so an upgrade can be made before deciding to implement this feature.
- One of the advantages of using *Javascript* for this first version of T-CREo is the fast implementation; however, it does not scale very well on productive environments. We can port the code to a different programming language that supports threads and has better performance.

[16]https://realtimeapi.io/hub/realtime-api-design-guide/

[17]Tested by Robert Miller https://humanbenchmark.com/

[18]https://www.nginx.com/

[19]https://nodejs.org/dist./v10.16.0/docs/api/worker_threads.html

TABLE 6. Time Per Request at `/Calculate/Tweet/Scraped` **in ms**

Requests (-n)	Concurrency (-c)	Scenario a		Scenario b		Scenario c	
		Local SF Low	**Local SF High**	**SF Low SF Low**	**SF High SF High**	**SF Low NY Low**	**SF Low NY High**
50	5	2.24	2.85	4.55	1.84	83.88	32.99
	10	2.42	1.95	3.90	1.53	35.60	18.35
	20	2.33	1.86	3,74	1.45	22.07	12.56
	30	1.98	1.70	4.10	1.48	18.0	11.58
100	5	2.06	1.65	3.11	1.38	40.11	30.41
	10	2.08	1.51	3.90	1.30	23.27	16.19
	20	1.85	1.40	3.18	1.31	16.74	9.59
	30	1.66	1.49	2.72	1.28	14.42	8.01
200	5	1.87	2.47	10.01	2.43	35.85	29.65
	10	1.87	1.32	3.43	1.34	34.40	15.35
	20	1.79	1.30	2.88	1.26	14.08	9.62
	30	1.84	1.31	3.21	1.29	12.77	6.50
500	5	1.66	1.34	6.55	1.88	34.83	29.06
	10	2.95	1.32	3.10	1.23	20.26	16.01
	20	3.15	1.42	3.04	1.25	11.54	7.69
	30	1.41	1.69	3.13	1.24	16.91	5.82
1000	5	1.70	1.33	5.16	1.50	34.90	28.60
	10	1.48	1.55	3.10	1.22	19.33	14.78
	20	1.35	1.25	3.22	1.25	13.53	7.83
	30	1.41	1.29	4.44	1.52	10.29	5.19
2000	5	1.47	1.40	3.94	1.38	34.81	28.79
	10	1.17	1.39	3.87	1.35	19.64	14.47
	20	1.97	1.41	3.09	1,25	12.02	7.30
	30	1.15	1.42	4.44	1.39	12.79	5.03
AVG		**1.87**	**1.57**	3.99	**1.43**	**24.67**	15.47
MIN		1.15	1.25	2.72	1.22	10.29	5.03
MAX		3.15	2.85	10.01	2.43	**83.88**	32.99

The current implementation is well modularized and has a few quantity of functions (roughly 2000 lines of code and mostly of configuration); thus, porting T-CREo to another programming language would not take a lot of effort.

Another idea to improve the performance is to use a search engine or an external database to implement some of the components of the data provider layers from Figure 3. On the current implementation, the data providers layer runs in the same process of the T-CREo back-end, therefore the dictionary lookup algorithms are implemented in *Javascript*, which is not ideal for highly intensive CPU operations. This also makes the server to take some time (aprox. 1 minute) to start and be able to receive petitions. By delegating these lookups for *misspelling* and *bad words* filters in a separate application, such as Elasticsearch[20] or even a simple RDBMS, like PostgreSQL, the server can be able to receive petitions earlier and the lookups can be made in a most efficient way.

[20]https://www.elastic.co/es/what-is/elasticsearch

B. EXPERIMENTAL EVALUATION OF THE TIME COMPLEXITY

Regarding the evaluation of the credibility model, it was not worth doing experiments to demonstrate its $O(n)$ time complexity, with n representing the number of words in the analysed text and being bounded by the *Text Credibility* (see Section III-E). Since the maximum number of characters in Twitter is quite short, it does not represent an appreciable time. However, in other social media without text size limitation, the *Text Credibility* impact can be better measured.

C. EXTEND THE IMPLEMENTATION FOR OTHER SOCIAL NETWORK PLATFORMS

Although the current version of T-CREo framework only supports Twitter, the proposed architecture works for any social network that we can either run web scraping or has an API. This flexibility allows integrating other social media networks into T-CREo. Integrating other three social networks, such as LinkedIn, Reddit, and Facebook, is also feasible.

LinkedIn is an employment-oriented social network, mainly used for professional networking. There are several kinds of user, but the most important (and the ones we focus)

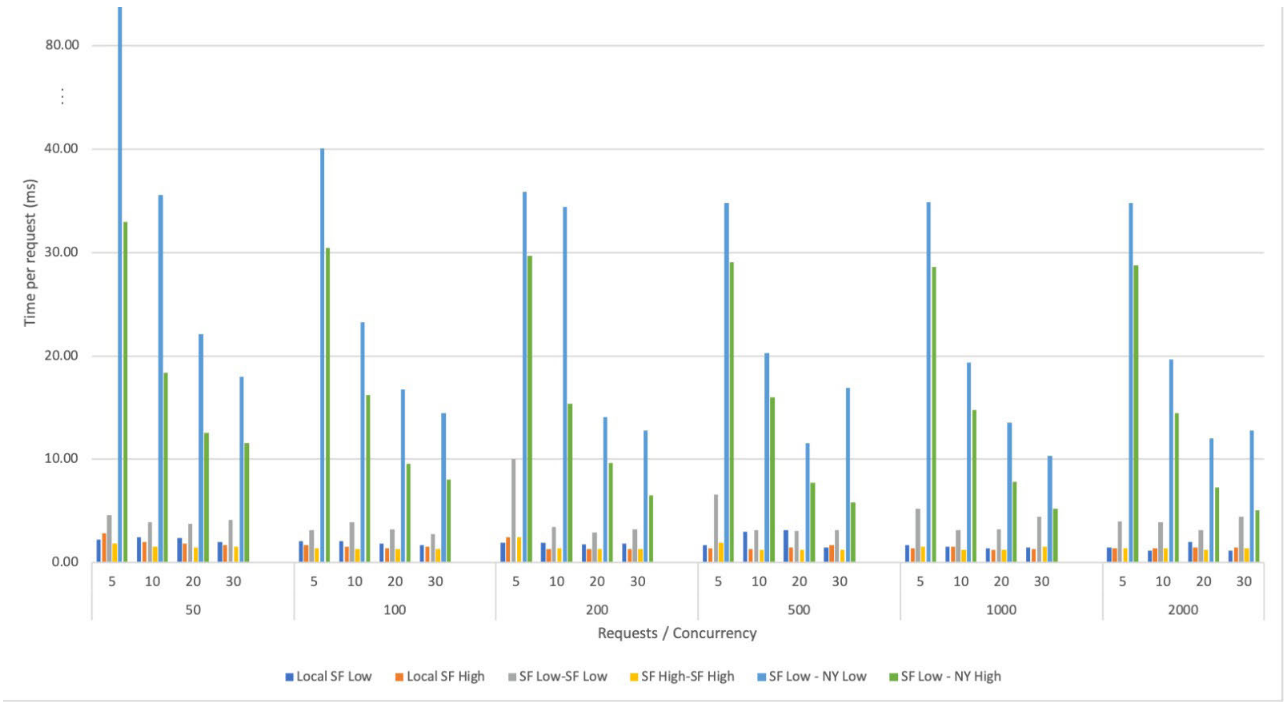


FIGURE 9. Time per request from Table 6.

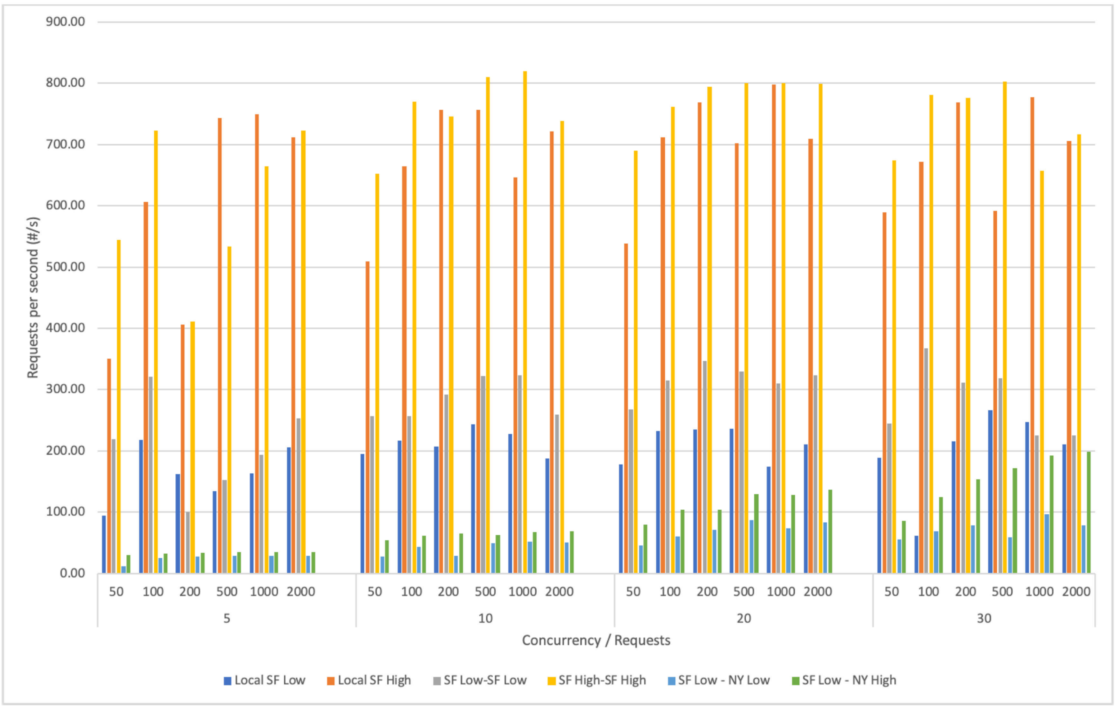


FIGURE 10. Requests per second from Table 7.

is the regular user type. These users usually are job seekers and company recruiters. Adding support for LinkedIn in T-CREo can be done via web scraping on user's activity feed, similar to how it is implemented for Twitter users' profile. That section can be visited by going to a user's profile and clicking ''See All'' on the Activity section. There, we can scrap each post, that contains text and other attributes that can enhance the model. The analogy of a ''verified user'' from Twitter can be ported to a ''premium user'' on LinkedIn. The only attribute that is not available to scrap is the creation year of the account, but it can be changed by the oldest year in their work experiences. LinkedIn have an API, but

TABLE 7. Number of Requests Per Second at `/Calculate/Tweet/Scraped`

Requests (-n)	Concurrency (-c)	Scenario a: Local SF Low	Scenario a: Local SF High	Scenario b: SF Low / SF Low	Scenario b: SF High / SF High	Scenario c: SF Low / NY Low	Scenario c: SF Low / NY High
50	5	446.46	350.29	219.78	543.99	11.90	30.30
	10	413.24	509.49	256.46	652.11	28.10	54.50
	20	429.20	538.24	267.75	689.75	45.40	79.60
	30	505.08	589.72	244.21	673.88	55.50	86.40
100	5	485.47	606.70	321.22	723.32	24.95	32.90
	10	480.80	664.31	256.51	769.57	43.00	61.80
	20	540.56	712.15	314.84	761.16	59.80	104.40
	30	602.42	671.60	367.64	781.60	69.30	124.80
200	5	534.77	405.62	99.90	411.09	27.90	33.75
	10	534.78	756.85	291.88	745.62	29.10	65.00
	20	558.67	768.80	347.15	794.40	71.00	104.00
	30	543.52	768.80	311.74	776.00	78.30	153.60
500	5	602.45	743.85	152.76	533.15	28.70	34.40
	10	339.02	756.64	322.49	810.45	49.40	62.40
	20	317.48	702.03	329.10	800.19	86.60	130.00
	30	709.24	591.82	319.29	803.47	59.10	171.60
1000	5	588.25	749.18	193.70	664.59	28.65	34.95
	10	675.71	646.01	323.07	819.67	51.70	67.70
	20	740.76	798.36	310.45	800.51	74.00	127.80
	30	709.23	777.60	225.44	657.78	97.20	192.60
2000	5	680.28	712.32	253.62	722.49	28.70	34.75
	10	854.71	722.13	258.72	739.08	50.90	69.10
	20	507.63	709.42	323.16	799.77	83.20	137.00
	30	869.57	706.29	225.35	717.20	78.30	198.60
AVG		**569.55**	**664.93**	272.34	**716.29**	**52.53**	91.33
MIN		317.48	350.29	99.90	411.09	11.90	30.30
MAX		869.57	798.36	367.64	819.67	97.20	198.60

it is very limited. We need to request permission to users to access their information, unlike Twitter, that we can use their API token to access any user's information without their permission.

Reddit describes themselves as "the front page of the Internet". It is a social network where users share posts in sub-forums on the platform, each sub-forum is focused on a topic and it is maintained by users themselves. Applying web scraping on Reddit is a hard task since the HTML contents of the page are generated automatically by a third-party software, but their API offers the necessary resources for an integration on T-CREo to be possible. The credibility model might need to be modified, since some fields are not available on Reddit, or have another meaning. For example, there is a *following* and *followers* notion, but another important metric for social credibility is *karma*, which is a number that increases as the user is more active in communities and makes good contributions.

On Facebook, we cannot do web scraping because of the same reasons of Reddit. The CSS selectors used for scraping are randomly generated on each new release and changes very frequently. They have an API, but we need to explicitly request permission to other users to access their information, as on LinkedIn. These problems make it difficult to make an integration on both available choices with Facebook, although it is not impossible to implement. In scenarios in which sources must be trustworthy and reliable, such as journalist and governor accounts, it should be a must for them to grant permission for credibility analysis (in an ideal world!).

D. CREDIBILITY ANALYSIS OF TEXT ON IMAGES

Another enhancement for T-CREo is to read text from images and run our credibility model with that text. This can be easily done by running *Optical Character Recognition*[21] (OCR) software on the back-end. The integration can be done in a:

- Dedicated *endpoint*, say `/image`, that receives the same parameters from `/plain-text` but instead of a `text` it can receive an image URL or a sequence of bytes that represents the image.
- Enhancing our Twitter API integration to get all images of a tweet and include the text found in any of the images in the tweet's text.

Although we do not have experience implementing or using OCR libraries on Node.js, a search of

[21] https://en.wikipedia.org/wiki/Optical_character_recognition

`Optical Character Recognition` and `OCR` on `npm` shows that all popular choices are bindings of *Tesseract*,[22] which is an OCR engine open-sourced by HP. `node-tesseract-ocr`[23] seems like a good choice for this, since its bindings are really similar to the bindings of the original *Tesseract* implementation.

E. REDUNDANCY AS A STRATEGY TO OVERCOME SOCIAL MEDIA API LIMITATIONS

In the studied social media, Twitter is the one that has the most flexible permissions, but it still has some limitations on the number of requests that we can consume in a time window. This limitation is not exclusive of Twitter API. A solution for this problem is to keep a shallow copy of the requested tweet in a database that T-CREo's back-end can query and synchronize regularly with Twitter real data. A synchronization strategy can be to run a scheduled work until all data is updated. The *endpoint* to get tweets' detail has a certain number of requests per 15 minutes time window. We can implement a job that updates as much tweets as it can and updates the remaining tweets of our internal database 15 minutes later.

This would not only overcome the problem with request usage limits, but can also be used to improve the credibility model from [4], by storing more properties and data.

VII. CONCLUSION AND FUTURE WORK

In this work, we propose a general architecture of a framework for credibility analysis in social media based on a general credibility model. The framework is capable of calculating credibility on any social media in real-time, combining web-scraping and social media APIs to gather the parameters needed to instantiate the credibility model. A proof of concept, for a specific use case of Twitter and to show the feasibility of the proposed architecture, named T-CREo (**T**witter **CRE**dibility analysis framew**o**rk), is developed and tested to evaluate its performance. Results show that our proposed framework can be implemented as a real-time service and the scalability is ensured by increasing the level of concurrency. This experience allows outlining some suggestions to improve overall performance for high-capacity servers. The modularity and simplicity of T-CREo, and the use of the credibility model, enable the creation of a real-time service; however, the connection time (latency) can be a determining factor, that might be considered in the deployment of the system.

Our future research is focused on the improvement of T-CREo, starting with the suggestions from Section VI, such as the implementation of several instances or multi-threaded versions of the back-end to improve the performance, keep an external database of posts to overcome API limitations, and incorporate credibility analysis in other social platforms, to provide a robust architecture to the community for the development of third-party applications. We also plan to extend the credibility model by considering bots detection, semantic analysis of the text, and multimedia data analysis.

[22]https://github.com/tesseract-ocr/tesseract

[23]https://www.npmjs.com/package/node-tesseract-ocr

ACKNOWLEDGMENT

The authors would like to thank Sergio Barrios, Fabiola Martínez, Yuni Quintero, José Acevedo, and Nairelys Hernández who have contributed to the implementation and experiments.

REFERENCES

[1] D. Westerman, P. R. Spence, and B. Van Der Heide, ''Social media as information source: Recency of updates and credibility of information,'' *J. Comput.-Med. Commun.*, vol. 19, no. 2, pp. 171–183, Jan. 2014.

[2] Y. Kammerer, E. Kalbfell, and P. Gerjets, ''Is this information source commercially biased? How contradictions between Web pages stimulate the consideration of source information,'' *Discourse Process.*, vol. 53, nos. 5–6, pp. 430–456, Jul. 2016.

[3] J. Slomian, O. Bruyère, J. Y. Reginster, and P. Emonts, ''The Internet as a source of information used by women after childbirth to meet their need for information: A Web-based survey,'' *Midwifery*, vol. 48, pp. 46–52, May 2017.

[4] I. Dongo, Y. Cardinale, and A. Aguilera, ''Credibility analysis for available information sources on the Web: A review and a contribution,'' in *Proc. 4th Int. Conf. Syst. Rel. Saf. (ICSRS)*, Nov. 2019, pp. 116–125.

[5] S. Y. Rieh and D. R. Danielson, ''Credibility: A multidisciplinary framework,'' *Annu. Rev. Inf. Sci. Technol.*, vol. 41, no. 1, pp. 307–364, 2007.

[6] T. J. Johnson and B. K. Kaye, ''Reasons to believe: Influence of credibility on motivations for using social networks,'' *Comput. Hum. Behav.*, vol. 50, pp. 544–555, Sep. 2015.

[7] Omnicore. (2020). *Twitter by the Numbers: Stats, Demographics & Fun Facts*. [Online]. Available: https://www.omnicoreagency.com/twitter-statistics

[8] C. Castillo, M. Mendoza, and B. Poblete, ''Information credibility on Twitter,'' in *Proc. 20th Int. Conf. World Wide Web*, 2011, pp. 675–684.

[9] B. Kang, T. Höllerer, and J. O'Donovan, ''Believe it or not? Analyzing information credibility in microblogs,'' in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, Aug. 2015, pp. 611–616.

[10] S. M. Shariff, X. Zhang, and M. Sanderson, ''On the credibility perception of news on Twitter: Readers, topics and features,'' *Comput. Hum. Behav.*, vol. 75, pp. 785–796, Oct. 2017.

[11] M. Alrubaian, M. Al-Qurishi, A. Alamri, M. Al-Rakhami, M. M. Hassan, and G. Fortino, ''Credibility in online social networks: A survey,'' *IEEE Access*, vol. 7, pp. 2828–2855, 2019.

[12] M. Viviani and G. Pasi, ''Credibility in social media: Opinions, news, and health information—A survey,'' *Wiley Interdiscipl. Rev., Data Mining Knowl. Discovery*, vol. 7, no. 5, p. e1209, Sep. 2017.

[13] H. S. Al-Khalifa and R. M. Al-Eidan, ''An experimental system for measuring the credibility of news content in Twitter,'' *Int. J. Web Inf. Syst.*, vol. 7, no. 2, pp. 130–151, Jun. 2011.

[14] A. Gupta, P. Kumaraguru, C. Castillo, and P. Meier, ''Tweetcred: Real-time credibility assessment of content on Twitter,'' in *Proc. Internat. Conf. Social Informat.*, 2014, pp. 228–243.

[15] M. AlRubaian, M. Al-Qurishi, M. Al-Rakhami, M. M. Hassan, and A. Alamri, ''CredFinder: A real-time tweets credibility assessing system,'' in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, Aug. 2016, pp. 1406–1409.

[16] T. Stephanie, ''Spot the lie: Detecting untruthful online opinion on twitter,'' Ph.D. dissertation, Dept. Comput., Imperial College London, London, U.K., 2017. [Online]. Available: https://www.doc.ic.ac.uk/~oc511/reportStephanie.pdf

[17] J. Yang, M. Yu, H. Qin, M. Lu, and C. Yang, ''A twitter data credibility framework—Hurricane Harvey as a use case,'' *ISPRS Int. J. Geo-Inf.*, vol. 8, no. 3, p. 111, Feb. 2019.

[18] A. Iftene, D. Gîfu, A.-R. Miron, and M.-S. Dudu, ''A real-time system for credibility on Twitter,'' in *Proc. The 12th Lang. Resour. Eval. Conf.*, 2020, pp. 6166–6173.

[19] K. R. Saikaew and C. Noyunsan, ''Features for measuring credibility on facebook information,'' *Int. Scholarly Sci. Res. Innov.*, vol. 9, no. 1, pp. 174–177, 2015.

[20] A. M. Idrees, F. Kamal, and A. I., ''A proposed model for detecting facebook News' credibility,'' *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 7, pp. 311–316, 2019.

[21] A. Black, C. Mascaro, M. Gallagher, and S. P. Goggins, "Twitter zombie: Architecture for capturing, socially transforming and analyzing the Twittersphere," in *Proc. 17th ACM Int. Conf. Supporting Group Work*, 2012, pp. 229–238.

[22] M. Congosto, P. Basanta-Val, and L. Sanchez-Fernandez, "T-hoarder: A framework to process Twitter data streams," *J. Netw. Comput. Appl.*, vol. 83, pp. 28–39, Apr. 2017.

[23] A. Hernandez-Suarez, G. Sanchez-Perez, K. Toscano-Medina, R. Toscano-Medina, V. Martinez-Hernandez, J. Olivares-Mercado, H. Pérez-Meana, and V. Sanchez, "Can Twitter API be bypassed? A new methodology for collecting chronological information without restrictions," in *Proc. SoMeT*, 2018, pp. 453–462.

[24] A. Hernandez-Suarez, G. Sanchez-Perez, K. Toscano-Medina, V. Martinez-Hernandez, V. Sanchez, and H. Pérez-Meana, "A Web scraping methodology for bypassing Twitter API restrictions," *CoRR*, vol. abs/1803.09875, pp. 1–8, Mar. 2018.

[25] D. Freelon, "Computational research in the post-API age," *Political Commun.*, vol. 35, no. 4, pp. 665–668, Oct. 2018.

[26] B. Kusumasari and N. P. A. Prabowo, "Scraping social media data for disaster communication: How the pattern of Twitter users affects disasters in Asia and the pacific," *Natural Hazards*, vol. 103, no. 3, pp. 3415–3435, Sep. 2020.

[27] I. Dongo, Y. Cadinale, A. Aguilera, F. Martínez, Y. Quintero, and S. Barrios, "Web scraping versus Twitter API: A comparison for a credibility analysis," in *Proc. 22nd Int. Conf. Inf. Integr. Web-Based Appl. Services*, Nov. 2020, pp. 1–11.

[28] O. Goonetilleke, T. Sellis, X. Zhang, and S. Sathe, "Twitter analytics: A big data management perspective," *ACM SIGKDD Explor. Newslett.*, vol. 16, no. 1, pp. 11–20, 2014.

[29] R. Giovanetti and L. Lancieri, "Model of computer architecture for online social networks flexible data analysis: The case of Twitter data," in *Proc. IEEE/ACM Internat. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, Aug. 2016, pp. 677–684.

[30] M. Fu, A. Agrawal, A. Floratou, B. Graham, A. Jorgensen, M. Li, N. Lu, K. Ramasamy, S. Rao, and C. Wang, "Twitter heron: Towards extensible streaming engines," in *Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE)*, Apr. 2017, pp. 1165–1172.

[31] J. Jarrett, K. Hemmings-Jarrett, and M. B. Blake, "Towards a service-oriented architecture for pre-processing crowd-sourced sentiment from Twitter," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2019, pp. 163–171.

[32] Y. Namihira, N. Segawa, Y. Ikegami, K. Kawai, T. Kawabe, and S. Tsuruta, "High precision credibility analysis of information on Twitter," in *Proc. Int. Conf. Signal-Image Technol. Internet-Based Syst.*, Dec. 2013, pp. 909–915.

[33] C. Shao, G. L. Ciampaglia, A. Flammini, and F. Menczer, "Hoaxy: A platform for tracking online misinformation," in *Proc. 25th Int. Conf. Companion World Wide Web*, 2016, pp. 745–750.

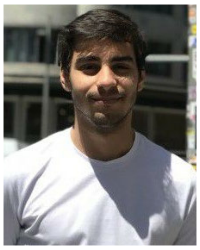
[34] T. Hamdi, H. Slimi, I. Bounhas, and Y. Slimani, "A hybrid approach for fake news detection in Twitter based on user features and graph embedding," in *Proc. Int. Conf. Distrib. Comput. Internet Technol.* Bhubaneswar, India: Springer, 2020, pp. 266–280.

[35] I. Dongo, Y. Cardinale, and R. Chbeir, "RDF-F: RDF datatype inFerring framework," *Data Sci. Eng.*, vol. 3, no. 2, pp. 115–135, Jun. 2018.

YUDITH CARDINALE received the degree (Hons.) in computer engineering from Universidad Centro-Occidental Lisandro Alvarado, Venezuela, in 1990, and the M.Sc. and Ph.D. degrees in computer science from Universidad Simón Bolívar (USB), Venezuela, in 1993 and 2004, respectively. She has been a Full Professor with the Computer Science Department, USB, since 1996. She is currently an Associate Researcher with the Universidad Católica San Pablo, Arequipa, Peru. She has written a range of scientific articles published in international journal, books, and conferences, and has participated as a member of program committees of several international conferences and journals. Her research interests include parallel processing, distributed object processing, operating systems, digital ecosystems, high performance on grid and cloud platforms, collaborative frameworks, and web services composition, including semantic web.

IRVIN DONGO received the B.Sc. degree in computer science from Universidad Católica San Pablo, Peru, in 2012, and the M.Sc. and Ph.D. degrees from the University of Pau, France, in 2014 and 2017, respectively. He was a Postdoctoral Fellow with the École Superieure des Technologies Industrielles Avancées (ESTIA) Institute of Technology, France, from 2018 to 2019. He is currently an Associate Researcher in computer science with ESTIA Institute of Technology and also with Universidad Católica San Pablo. His research interests include the normalization and anonymization of Web resources, knowledge-bases modeling (Semantic Web), policies and management of credentials, security model and anonymization technique, and machine/deep learning techniques for an analysis and classification of data to discover patters and gesture recognition.

GERMÁN ROBAYO is currently pursuing the degree and the bachelor's degree in computer engineering from Universidad Simón Bolívar. Through his career he has developed software architecture skills, by enrolling in several courses related to that subject. His research interests include programming languages design and machine learning, specifically in the field of natural language processing.



DAVID CABEZA is currently pursuing the degree with Universidad Simón Bolívar. He is also a Computer Engineer with Universidad Simón Bolívar. Throughout his career, he has developed skills in algorithms, software engineering, among others. His research interests include conducting research in cryptography, software engineering, and the intersection between business, technology, and user experience.

ANA AGUILERA received the B.S. degree in computer science engineering from Lisandro Alvarado West-Central University (UCLA), Barquisimeto, Venezuela, in 1994, the B.Sc. degree (Hons.) in computer engineering from the University of Rennes I, Rennes, France, the M.S. degree in computer science from Universidad Simón Bolívar, Caracas, Venezuela, in 1998, and the Ph.D. degree in medical informatics from the University of Rennes I, in 2008. She is currently a Full Professor with the Faculty of Engineering, Escuela de Ingeniería Informática, University of Valparaíso, Valparaíso, Chile. Her research interests include fuzzy databases, data mining, social networks, and medical informatics. Since 2011, she has been a member of the Program Encouragement for Research and Innovation Researcher (PEII) Level C, Venezuela. She is a member of the Venezuelan Association for the Advancement of Science (AsoVAC) and of the Venezuelan Computer Society (SVC). She received the Magna Cum Laude Award from UCLA and "Très honorable" Award in the Ph.D. thesis from Rennes I. She was accredited in Program for Researcher Promotion of Venezuela, Candidate Level, in 1998.

SERGIO MEDINA received the bachelor's degree in computer engineering from Universidad Simón Bolívar (USB), Sartenejas, Caracas. He is currently a Software Developer with Rappi S.A.S, Bogotá, Colombia. His research interests include software architecture, event driven, and data intensive designs.

• • •