Everyone's Invited: A New Paradigm For Evaluation on Non-transferable Datasets

David Jurgens, Tyler Finnethy, Caitrin Armstrong, and Derek Ruths
jurgens@cs.mcgill.ca, druths@networkdynamics.org,
{tyler.finnethy,caitrin.armstrong}@mail.mcgill.ca
School of Computer Science
McGill University

Abstract

Social media data mining and analytics has stimulated a wide array of computational research. Traditionally, individual researchers are responsible for acquiring and managing their own datasets. However, the temporal nature of social data, the challenges involved in correctly preparing a dataset, the sheer scale of many datasets, and the proprietary nature of many data sources can make extending and comparing computational methods difficult and often impossible. In light of this, because replicability is a fundamental pillar of the scientific process and because method comparison is essential to characterizing computational advancements, we require an alternative to the traditional model of researcher-owned datasets. In this paper we propose FREESR, a framework that gives researchers the ability to develop and test method performance without requiring direct access to "shared" datasets. As a case study and first community resource, we have implemented the FREESR paradigm around the task of Tweet geolocation. The implementation showcases the clear suitability of this framework for the social media research context. Beyond the implementation, we see the FREESR paradigm as being an important step towards making study reproducibility and method comparison more principled and ubiquitous in the social media research community.

1 Introduction

When developing methods that run on social media data, datasets are used to both establish the performance of one's method and also to compare it to existing alternatives, ideally highlighting where advances have been made. Moreover, reproducing existing studies using their datasets is an important means for the community to understand and characterize the validity (and implicit assumptions) made by the originators of the work. In both these cases, shared datasets serve an essential role and in many domains, foster comparative work on a single research objective.

The growth in social media data has led to many new research questions and corresponding datasets have been created for developing computational models in pursuit of these questions. However, the proprietary nature of social media leads to two key problems not traditionally faced when evaluating: (1) the **Data Replicability** problem, where privacy restrictions or Terms of Service prohibit the direct sharing of datasets between researchers, and (2) the **Data Decay** problem where items in the original dataset are removed by users as time passes, further exacerbating the replication problem (McCreadie et al. 2012; Almuhimedi et al. 2013). While social media data has still been used for evaluation tasks within the research community (Ounis et al. 2011; Nakov et al. 2013), research efforts have been hindered by both problems, with the resulting datasets size-constrained by the limited ability to reconstruct the original data and with decay masking the initial state of the data.

To overcome both the Data Sharing and Data Replicability problems, we present a new evaluation paradigm that allows replicable evaluation on non-transferable datasets. In contrast to the traditional setting where datasets are moved to the location of computational methods and tested locally, in our setting, methods are moved to the datasets themselves, which are accessible through a secure API. The method is then evaluated remotely and its evaluation results are shared back with the experimenter. By allowing full access to the data under controlled conditions that prevent outside data transmission, this approach solves the Data Replicability problem while staying within the data's terms of use. Furthermore, because computational models may be stored remotely with the data, this potentially solves the Data Decay problem by updating the reported performance of each model as the underlying data changes over time. We refer to this paradigm as the **F**ramework for **R**eproducible **E**valuation of **E**xperiments with **S**ensitive **R**esources, abbreviated as FREESR.

To quantify the benefits and challenges of FREESR, we present a case study of implementing a shared real-world task facing the Data Sharing and Data Decay problems. Our work offers three main contributions. First, we define the FREESR paradigm and both describe how it solves the data replicability problem currently impacting social media evaluations and outline the key challenges for a successful task design. Second, we describe the process of creating a FREESR-based task for geolocation inference, which aims to infer the location of social media posts. The resulting hosted task is made publicly-available for the research community and addresses a significant need for comparability between current and future work. Furthermore, software for the platform is available as open source, thereby mak-

ing it easier for new FREESR-based tasks to be developed by other researchers with sensitive data. Third, we identify how the core challenges for developing a FREESR task were addressed during the development process and identify open questions that need further improvement for robust evaluation practices.

2 Related Work

The proposed FREESR model builds upon prior work in developing shared tasks, replicable evaluation design. and the creation of social media datasets.

Shared Evaluation Settings Shared tasks have served an important role in the research community by creating high-quality data and annotations for evaluation and by attracting community interest to particular research problem. The shared task paradigm has seen wide-spread adoption with standing evaluation conferences in many research areas, such as Natural Language Processing with Senseval (Kilgarriff and Palmer 2000), Information Retrieval with CLEF (Braschler and Peters 2004) and TREC (Voorhees, Harman, and others 2005), and Information Extraction with MUC (Grishman and Sundheim 1996). Indeed, even full conferences have included annual shared tasks to encourage work on a particular problem (Tjong Kim Sang and Buchholz 2000; Morante and Blanco 2012) or have developed regular dataset sharing initiatives to encourage replicability, such as that seen for ICWSM.[1] Nevertheless, nearly all of these tasks have evaluated using transferable datasets, which do not pose the same replication challenges as social media datasets.

Remote evaluation While the research community has largely adopted public test sets for evaluating new approaches, several competitive programming platforms such as CodeChef[2] and HackerRank[3] have incorporated hidden test sets. These competitions focus on a computational task, such as computing the greatest common divisor in the minimum time. Each competition defines (a) the input format of the data, representing a particular instance of the task to be solved and (b) the expected output format of the program's solution to an input instance. While some examples are shown publicly to let competitors test their solution, to compete on these platforms, developers submit a program, which is subsequently tested on unseen input that is never released.

Twitter Datsets Given the utility of Twitter data, multiple corpora have been proposed as community resources for tasks such as sentiment analysis (Pak and Paroubek 2010), event detection (McMinn, Moshfeghi, and Jose 2013), and geolocation inference (Eisenstein et al. 2010; Mahmud, Nichols, and Drews 2012). Initial attempts to create publicly-sharable corpora from anonymized Twitter data (as opposed to post IDs) resulted in the corpus being removed due to violating terms of service (Petrovic, Osborne, and Lavrenko 2010); as a result, all shared datasets are released post IDs that must be used in conjunction with the Twitter API to retrieve the dataset's content. These post ID datasets still encounter the Data Decay problem since the underlying posts may be deleted and also are constrained by the Twitter API rate limits, which effectively limits how many posts can be feasibly obtained by researchers.

Despite the challenges of using post ID-based datasets, several shared tasks incorporated Twitter data in this matter. Ounis et al. (2011) built a Twitter corpus as a part of a TREC shared task, but later note that up to 22% of posts had been removed within several months (McCreadie et al. 2012) . While their follow-up analysis showed that the deletions did not affect one system more than other of those that participated in the task, the high deletion rate underscores the challenge of supporting later evaluation after much of the data has been removed. Indeed, during organization of a task for sentiment analysis on Twitter, Rosenthal et al. (2014) report nearly half of annotated training data was deleted by the time the task began. Thus, a FREESR task using Twitter data can serve an import role by allow the community to perform experiments on more data than is feasible by the API-access alone and by being able to conduct experiments on identical data.

3 The FREESR Paradigm

The FREESR paradigm addresses the Data Sharing and Data Decay problems using a hosted evaluation environment that securely allows access to unseen data while providing full transparency in all other areas of evaluation. Before discussing the design choices involved in implementing a particular task using FREESR, we first formalize the general terminology for each part of the system and how the parts interact. Figure 1 shows the high-level diagram of each component and their interactions.

An **experimenter** is any individual intending to perform an experiment that tests the performance of a particular method on a non-distributable dataset. To conduct an experiment, the experimenter creates a **submission** that encapsulates the method as (a) executable software, (b) a list of software dependency requirements, such as machine learning or natural language processing libraries, and (c) other non-software resources needed by the method, such as gazetteers or word lists.

The submission is transmitted remotely through a user interface to a FREESR **host**, which carries out the experiment on behalf of the experimenter. The host assembles the pieces of the submission to create a **model** that receives access to the dataset as input and produces results for the FREESR-based task in a predefined format. Dataset access is mediated through an Application Program Interface (**API**), which defines what data is available and how it may be accessed and transformed.[4] The individual or organization responsi-

[1] http://icwsm.org/2015/datasets/datasets/

[2] http://www.codechef.com/

[3] http://www.hackerrank.com

[4] The API for a particular task may take on many forms beyond a software-based API, such a specification for a particular ordering of command-line arguments, predefined input file formats, or even

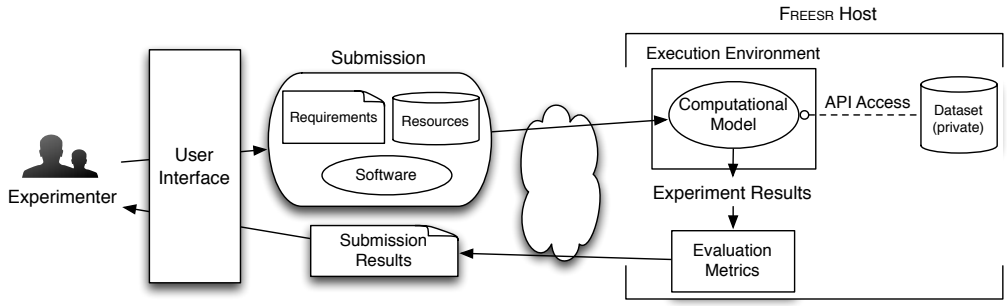


Figure 1: A diagram of the general FREESR testing model. An experimenter creates a submission of software and its resources, which is transferred to a remote host. The host assembles the submission's pieces into an computational model, which is then run in a secure execution environment that provides API-based access to the sensitive data. The model's results are then transmitted back to the experimenter.

ble for hosting a particular FREESR-based task is referred to as its **operator**.

The experiment itself is conducted within an **execution environment** which defines and enforces which resources and parts of the host system a computational model may access during its execution. The results of the computational model are then scored according to one or more evaluation metrics. The evaluation scores are returned to the experimenter through a user interface.

4 Challenges

The decoupling of experimenter from experiment and the remote execution of arbitrary computational models introduce multiple challenges for the FREESR paradigm, both for experimenter and operator. Following, we define the main challenges.

Usability

The FREESR paradigm requires that any computational method used in an experiment be designed according to a fixed interface for accessing the sensitive dataset. However, requiring a particular interface introduces constraints on the method's design and the ability of an experimenter to adapt their methodology to the host's requirements. Three usability challenges follow from the API requirements of a FREESR-based task.

Challenge 1 - Experimental Design: Accessing data solely through a fixed API or similar interface can potentially limit the design of a method or how the method can effectively use the data. For example, if a software interface is used to enforce an API, this requires all methods be implemented in the programming language of the API and also that all dependencies of the method be available in that language. The structure of the API may also introduce constraints and inconveniences if experimenters are required to subclass and include multiple predefined functions.

Challenge 2 - Data Access: Many experimenters refine large datasets to only the pertinent data; for example, experiments with Twitter commonly filter data to only those posts written in English, which can remove over half of all posts (Liu, Kliman-Silver, and Mislove 2014). Because experimenters cannot filter the data to match their needs ahead of run time, either the method must include additional functionality to perform this filtering, thereby performing needless computation for methods that aggressively filter, or the framework itself must offer different views of the same dataset, depending on method needs. The usability issue of filtering or otherwise transforming the dataset is compounded by repeated runs of methods that perform the same type of filtering, which would only have needed to be performed once if the dataset was available locally.

Challenge 3 - Interface Stability: Once an API is defined for a particular FREESR task, any changes to the API can become a problem. If experimenters have designed a submission based on now depreciated features, they may have to significantly rework their design to run on the updated framework. Although the framework operators will often not anticipate significant changes, unpredictable advancements within the field or security risks can force a reworking of the API. Furthermore, if the FREESR host supports re-running models as the data decays in order to reflect the current performance, non-compatible API changes may break its ability to re-run models that have not been updated to the new API.

Privacy

The requirement that experimental models be submitted to a host introduces three privacy-related challenges.

Challenge 4 - Intellectual Property: Communicating research software to a remote server raises the possibility that an adversarial operator of a FREESR-based task could steal an experimenter's intellectual property. Conducting research with computational methods is often a long-term process, with hypotheses being proposed and evaluated in an effort to maximize performance on the task at hand. Platforms such as `Arxiv.org` provide an important open-access outlet for in-progress research by allowing papers and results to be presented to the research community prior to peer review. However, no comparable platform hosts in-progress computational methods themselves while they are being tested and

a connection to databases that supports specific queries.

improved, potentially due to the fear of being scooped on an experimental result. A FREESR-based task raises the issue that if a novel method is submitted for evaluation, the yet-unpublished details of the approach could be stolen by an adversarial host for publication without credit to the original inventor.

Challenge 5 - Experimental Integrity: A similar privacy challenge centers on trust of the results. An adversarial operator with conflicting interests could alter the scores returned for a competitor's submissions. These lower scores could potentially discourage otherwise-successful research, or allow the adversarial experimenter to identify and develop upon successful a line of work before its first public disclosure.

Challenge 6 - Proprietary Resources: Beyond academia, research often occurs in industry and, in many cases, relies on proprietary software that cannot be distributed. While in the traditional evaluation setting, the software may be kept in-house and tested locally on the dataset, evaluation with a FREESR-based task would require that this proprietary software be uploaded to a third party host for remotely execution. Furthermore, in some legal jurisdictions, a new method may constitute intellectual property and therefore require specific actions be performed to secure its rights before its software could be shared. Thus, FREESR may impose a sharing requirement beyond what is feasible for industrial research to take part in, thereby creating a tiered system where only academic experimenters are able to test their methods.

Host Security

Hosting a FREESR-based task introduces two challenges due to the remote execution of arbitrary code.

Challenge 7 - Resource Integrity: The hosting platform represents a shared computing resource, which may be attacked by traditional adversarial means, such as deleting data, infecting the system with malware, or otherwise interfering with proper functioning of the operating environment. Furthermore, an adversarial experimenter may seek to monopolize the host by submitting an excessive number of experiments to run, thereby denying other researchers the ability to efficiently obtain results for their experiments.

Challenge 8 - Data Security: Beyond interfering with the hosting system, remotely executed code may violate the privacy of data and software on the system. Most importantly in the FREESR setting, an adversarial experimenter could leak the dataset, such as by directly communicating it through a new network connection or by encoding it as part of the results. Similarly, because the host may simultaneously be storing the experiments of other researchers, a malicious submission could potentially access and communicate other submissions' code, performing intellectual theft.

5 Geolocation Inference

Many social media analyses have used location meta-data to examine spatial phenomena such as well-being (Schwartz et al. 2013), language variation (Graham, Hale, and Gaffney 2014), topical discussions (Strnadova, Jurgens, and Lu 2013), and disaster response (Lingad, Karimi, and Yin 2013). While location data has proven useful, often very little social media data comes already annotated with the location of its origin. For example, less than 1% of Twitter posts come with GPS locations (Hecht et al. 2011). The task of geolocation inference, also known as *geoinference*, is to infer the location of the remaining posts that do not come with location, thereby providing a substantial boost in volume to subsequent analyses requiring spatially-located content.

Geoinference is an ideal task for FREESR for three reasons. First, the majority of approaches have been designed for and evaluated on Twitter data, which is proprietary and cannot be redistributed. Although the full datasets cannot be released, two geoinference datasets have been made publicly available as Twitter post IDs (Eisenstein et al. 2010; Mahmud, Nichols, and Drews 2012), thereby allowing independent reconstruction of the original data. However, due to the rate limitations in reconstructing data, these datasets are quite small compared to those used in recent work and also have been further reduced in size due to data decay. Indeed, releasing larger datasets for geoinference as Twitter post IDs is infeasible due to the practical limitations of accessing the original data through the Twitter API; for example, at current rate limits, recreating the dataset of Jurgens (2013) would take over 5,580 years.

A second motivation for selecting geoinference as a case study in FREESR task design is the need to test the feasibility of supporting the diversity of approaches. Geoinference techniques vary considerably in how locations are inferred, with some approaches using individual's social network in Twitter and others analyzing the text of an individual's posts for clues to their locations. Thus, the framework must be sufficiently general to support both the type of data access to each approach and the software dependencies and resources needed by each.

A third practical motivation for selecting geoinference is the current need for comparative evaluation between methods. Currently, geoinference methods have been tested in highly varied settings, with no standardization in evaluation metrics and with dataset sizes ranging from a few million posts (Li et al. 2012; Li, Wang, and Chang 2012; Rout et al. 2013) to hundreds of millions of posts or more (Jurgens 2013; Compton, Jurgens, and Allen 2014). As such, a full comparison between methods has been effectively prevented unless the computational models themselves are recreated and tested in one place in a setting that has access to comparable amounts of data as used in the original experiments, as was done in Jurgens et al. (2015).

6 A Geoinference Testing Framework

The architecture of the geoinference testing framework, shown in Figure 2, mirrors the FREESR design and has four main aspects: (1) the datasets used for testing, (2) the API and dependency specifications, (3) the hosting architecture, and (4) the user interface. Following, we describe how each aspect was implemented and how its design addresses the challenges posed when hosting a FREESRtask.

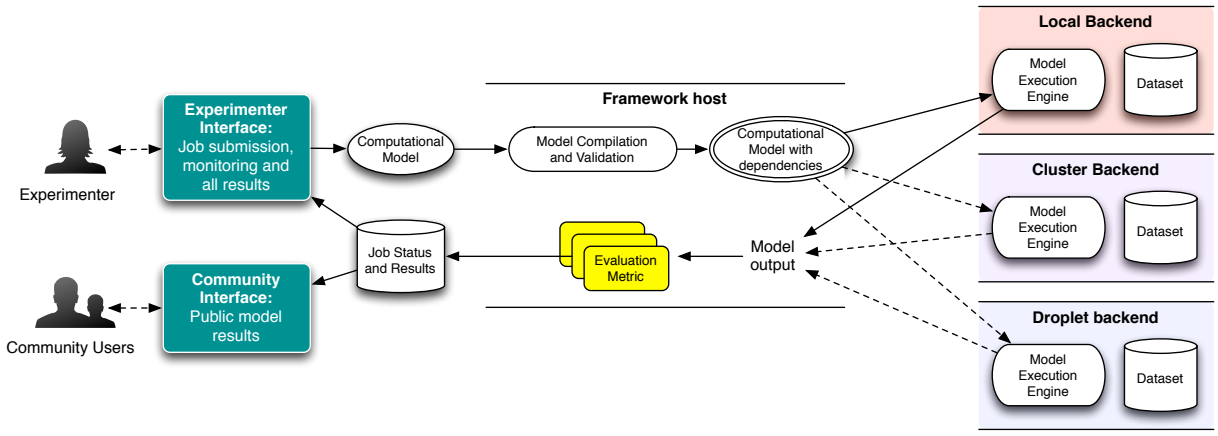


Figure 2: A diagram of the geoinference testing framework. Using a web interface, experimenters submit requests for new experimental runs. Each submission includes software for computational model and a specification of its software dependencies. This submission is then retrieved and validated by the hosting, which in turn schedules the experiment in an appropriate execution environment. Once the experiment is completed, the results are scored and communicated to the experimenter using a web interface, with an option to display the results publicly for the research community to see.

	Posts	Users	Bi-D. Mentions	Geolocated posts
9/14–9/14	1,243,328,563	72,021,244	26,429,346	1,245,913,305
8/14–9/14	2,478,859,385	94,328,017	55,821,290	2,482,449,903
7/14–9/14	3,918,859,653	118,856,427	81,422,484	3,923,193,848
6/14–9/14	5,190,703,686	138,918,914	106,887,695	5,196,291,219
5/14–9/14	6,457,017,352	155,334,765	130,459,422	6,464,384,438
4/14–9/14	7,657,048,348	168,993,318	153,220,866	7,665,185,573

Table 1: Summary of the six available datasets, with their relative sizes and the size of the social network extracted from bi-directional mentions within each.

Dataset

Prior geoinference testing settings have used datasets of significantly different sizes, in part due the disparity in access to large volumes of Twitter data. Therefore, in the proposed evaluation task, we selected datasets with two goals in mind: (1) to be representative of all Twitter data, and (2) to be of sufficient size that a method would be required to adapt its computational complexity to the scale required for real-world operations. Hence, the performance of any geoinference method on these datasets would be expected to generalize to new data.

Datasets were constructed from the Twitter gardenhose, which is roughly a 10% sample of all Twitter volume. The initial framework includes six official datasets, consisting of one to six months worth of Twitter data. Table 1 shows the size and properties of each. The single-month dataset is equivalent in size to the largest dataset used in geocoding, though in practice most approaches have aggressively filtered the data to a subset matching certain properties, such as those posts containing mentions or only those that have GPS data. Having a standardized dataset created from an query-independent sample of Twitter data potentially mitigates biases from using a dataset sampled from the Twitter API, which requires query parameters (e.g., hashtags or a geographic bounding box) and is known to not to be fully representative of all Twitter data (Morstatter et al. 2013).

We intentionally selected the smallest dataset to include a full month of data in order to provide a robust sample of the types of posts and cyclical volumes seen over a short-scale time period. We anticipate that a geoinference method would be expected to perform well given this much data, so the dataset reflects an ideal starting point for testing new methods. However, we do acknowledge that many geoinference methods have used smaller datasets, potentially due to their computational complexity. Therefore, the testing framework also includes unofficial datasets consisting of 1, 3, 7, and 14 days worth of posts from the September 2014 portion of the dataset in order to models to be designed and tested with increasingly-larger amounts of data.[5] Furthermore, the single-day dataset is sufficiently small to be completely recreated using Twitter's API, taking 26.5 days with current rate limits; because the framework is fully open sourced, having a recreated dataset enables researchers to validate the results of the host locally on identical data, which directly addresses Challenge 5.

Each dataset is partitioned into five folds that are used with cross-validation during evaluation. Partitions remain fixed, ensuring full comparability between submissions.

Geoinference API

Geoinference methods have typically adopted a variety of data for learning how to infer the location of a post. To effectively support current and future approaches, the API and its implementation were designed with three goals in mind: (1) the interface must place minimal requirements on how the method is to perform training or predict location, (2) all Twitter data provided as input should be made available in its

[5]In practice, the framework design allows methods to easily subsample from the full datasets, so using all posts is not strictly necessary.

```python
class GIMethod(object):
    """The abstract base class for all geolocation
       inference systems."""
    __metaclass__ = abc.ABCMeta

    @abc.abstractmethod
    def train_model(self,settings,dataset,model_dir):
        """ Builds a geoinference model using the settings
            and dataset provided and returns a trained GIModel."""
        pass

    @abc.abstractmethod
    def load_model(self,model_dir,settings):
        """Loads a subclass of `GIModel` from the
           directory model_dir"""
        pass

class GIModel(object):
    """The abstract base class for trained
       geoinference models."""
    __metaclass__ = abc.ABCMeta

    @abc.abstractmethod
    def infer_post_location(self,post):
        """Infers the lat-lon location for the
           post specified."""
        return

    def infer_posts_locations_by_user(self,user_id,posts):
        """This method infers the locations for each of
           the user's posts"""
```

(a) Python API Abstract Base Class

```java
public interface GIMethod {

    /**
     * Trains a new method from the provided dataset, returning
     * the trained model
     */
    GIModel train(JSONObject settings,
                  Dataset dataset, File modelDir);
    /**
     * Loads and returns a trained method from the
     * data in the directory
     */
    GIModel load(File modelDir);
}

public interface GIModel {

    /**
     * Infers the location of the provided post
     */
    LatLon inferPostLocation(JSONObject post);

    /**
     * Infers the location of each post for the provided user
     */
    List<LatLon> inferPostLocationsByUser(long userId,
                                          List<JSONObject> post);
}
```

(b) Java API Interface

Figure 3: The API specification for computational models in the geoinference testing framework.

original format,[6] thereby letting experimenters decide which portions should be used, and (3) commonly used features such as the twitter users' social network should be precomputed and made accessible through the API.

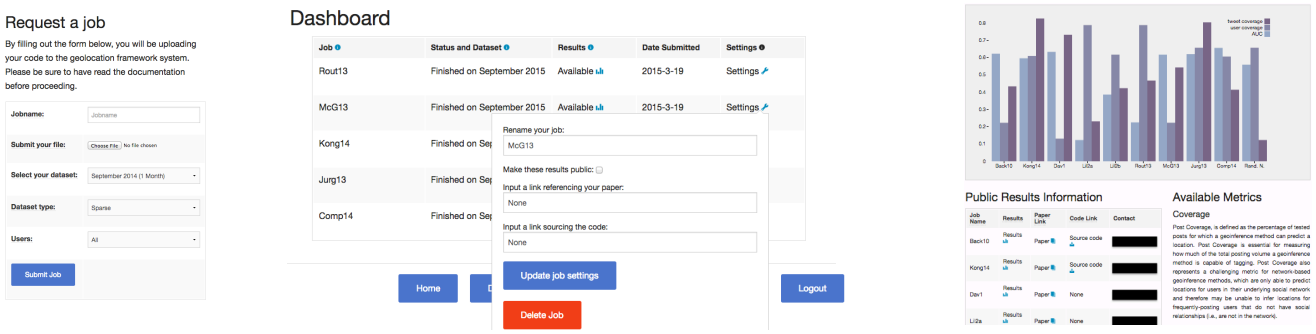
To satisfy the first goal, our API is designed in Python and Java, providing support for models in two of the most popular languages. The API requires that a computation model must either extend a Python abstract base class or implement a Java interface, both shown in Figure 3. The choice of supporting Python and Java was also strongly motivated by both languages' support for software dependency management. In our setting, Python programs may use `pip` and `easy_install` to automatically download and install public Python packages; similarity, Java programs may use Apache Maven to install libraries. Importantly, both language's dependency management suites cover the most widely-used libraries that will be needed for geoinference, such as SciKit (Pedregosa et al. 2011) and WEKA (Hall et al. 2009) for machine learning.

The interface in both languages is sufficiently general to capture the two main aspects of geoinference: training and location inference. The `GIMethod` class encapsulates the training portion of geoinference. Twitter data is accessed through the `Dataset` class, provided as an input parameter to the `train_model` method. Specifically, the `Dataset` class provides access to iterate through all posts arbitrarily and iterate through each user's posts, one at a time. Only iterative access is provided because the datasets are prohibitively large for use in-memory. The `Dataset` class also provides direct access to (a) different social networks underlying the data, such as all edges between two users mentioning each other, and (b) estimated locations for users based on the data they provide, such a self-reported location name in their profile. While this data could easily be computed by any `GIMethod` during its training stage, precomputing this data and providing it via the `Dataset` class can significantly reduce the computational run-time and potential for experimenter mistakes.

The `GIModel` class encapsulates the logic for inferring the location of a post. The `infer_post` method provides a single post for which the method must make a prediction. However, a user's posts may collectively provide clues to their location and thus we include the `infer_posts_locations_by_user` method to allow approaches to take advantage of all of a user's data at once. Despite the common goal of predicting location, geoinference methods vary in how the location is represented, which is commonly either a city name or a latitude-longitude point. Therefore, we designed both prediction methods to support the most general form of output as a latitude and longitude point. Geoinference methods that were originally designed to report a city name can simply report a canonical point denoting that city. This representation facilitates easy comparison between approaches and a point may always be converted to a location name via reverse geocoding.

The resulting API directly addresses Challenges 1, 2, and 3 for Usability. By designing a general-purpose API with minimal requirements for how geoinference is to be performed, the API places few limits on experimental design. Furthermore, by supporting the two of the most popular programming languages, each of which provide extensive external package management, the API does not restrict experimenters to a particular set of software dependencies. Second, the `Dataset` class provides access to many precom-

[6]The Twitter API provides access to posts as JSON objects, with full details at `https://dev.twitter.com/overview/api/tweets`.



(a) Job submission (b) Experimenter dashboard (c) Public results

Figure 4: The core pieces of the User Interface, which allows researchers to (a) submit new experiments, (b) monitor the progress and results of their current experiments, and for completed experiments, publicly release the experiment's results with links code and a paper, and (c) view publicly-released results on the official datasets.

puted results, such as the social network, which greatly reduces the burden on the experimenter for performing these routine tasks and potentially allows for faster design and implementation. Finally, by keeping the API to a minimal set of operations and re-using the original format of the data, the interface can remain stable while still supporting a variety of methods.

All software for the experimenter is available at `https://github.com/networkdynamics/geoinference-method-template`. To reduce the effort required to create a new method, the repository contains example implementations in each language showing how to perform basic operations for both text- and network-based geoinference methods.

Host Architecture

The host architecture is structured as three components: (1) a validation module that ensures the submission provided by the experimenter functions as a working geoinference method, (2) a backend service that is responsible for running the experiment in a secure environment and recording the model output, and (3) a series of evaluation methods that score the output. Underlying these three components is a centralized database that tracks the current state of the experiment, e.g., recording that the submission is currently running or has completed.

The host may potentially receive multiple submissions, necessitating a queue for running each experiment separately in order to guarantee sufficient execution resources. However, requiring experiments to wait in a queue before execution can potentially delay detecting easily-fixable issues, such as compilation problems, missing external dependencies, or incorrect output formatting. Together these experiences may discourage development and hinder the overall goal of easily experimental replication. Therefore, the validation module is included as a first step prior to the model being submitted to the queue in order to rapidly communicate back to the experimenter if a problem occurs when instantiating their model.

To account for the potential interest by the research community, the host implementation was designed with multiple execution environments. A local server is used for in-house testing and development, while large experiments and outside submissions by other experimenters are executed as jobs on a shared cluster. As a possibility for future expansion, we have also developed a prototype on DigitalOcean droplets,[7] which is a cloud hosting service that allows a new virtualized server instance to be created from an existing disk image; in our setting, the image already contains the Twitter datasets and only requires receiving the submission to run an experiment.

In our implementation, the validation module and backend services share the same functionality for running a computational model. First, the dependencies of the model are satisfied, ensuring that no external resources are needed by the model. Then, a secure virtual machine sandbox is started that prohibits all outside network access and enables write access only to a specified output directory. Specifically, the Docker platform is used to host a locked-down Linux virtual machine.[8] The software used by computational model is only ever executed within this sandbox to ensure host integrity. Finally, the output of the model is verified to ensure that no sensitive data is leaked.

This design satisfies Challenges 7 and 8 for Resource Integrity and Data Security. Because the model has extremely limited write permissions and is encapsulated in a virtual machine, the model's execution cannot affect the underlying host. Similarly, because each model is isolated in its own virtual machine, the software from other experimenters can never be accessed. Furthermore, because the model cannot establish a remote connect nor write arbitrary output data, the security of the sensitive data is guaranteed.

User Interface

The user interface defines how experimenters may interact with the host. Figure 4 shows the three key pieces of the interface. Once an experimenter logs into the interface,

[7] `https://www.digitalocean.com`
[8] `https://www.docker.com/`

they may submit new experiments as jobs to be executed (Fig. 4a). The experimenter selects which dataset is used and may further refine the dataset to only the specific subset needed by their method, e.g., only posts for those users in the social network, in order the avoid needless computation.

Each job is assigned a new unique name that allows the job and its results to be monitored on the Experimenter Dashboard (Fig. 4b). Experimental results include the scores produced by the evaluation metrics and a link to access a random sample of 10,000 twitter post IDs within the dataset, the locations inferred by the method for those posts, and the errors of the inferences. Because the full post content may be obtained from the Twitter API using the post IDs, this sample enables the experimenter to quickly recreate a small portion of the dataset and analyze their method's performance on specific posts.[9] Furthermore, the ability to access a sample of results directly addresses the Challenge 5; because the sample is representative of all results and because the evaluation methodology is publicly known, computing the evaluation metrics on the sample's predictions should serve as a close approximate to the scores reported on the user interface, thereby providing the experimenter some assurance that the host-reported results are accurate.

Once an experiment has completed, the experimenter can opt to make the results shared on the host's public results page (Fig. 4c). Each public result may be linked by the experimenter to an article or webpage describing the method and to a software repository hosting an implementation of the method. The public results page serves two essential purposes. First, the page maintains an accurate list of state of the art for each dataset, to which new results may be easily added as the field progresses. Second, because the underlying datasets undergo data decay (i.e., tweets are deleted), the page can be periodically updated with the new performance numbers for each system on the current data, which allows tracking the models' performance changes over time. Additionally, actively maintaining a full list of results creates an aspect of gamification, which may foster a sense of competition and encourage new approaches; and knowledge of which methods do well on some metrics and not on others can facilitate a comparison of method subtype strengths.

7 Discussion

The process of implementing a geoinference task in FREESR raises several open questions for the design of future tasks.

Performance The volume of social media gives rise to very large datasets. Normally, experimenters identify and extract the relevant portions of this data, thereby reducing experimentation time and improving performance. However, within FREESR, experimenters do not have access to the data, and within our implementation, experimenters cannot cache any memoized results or prepare an already-filtered version of the dataset to reduce the amount of data (e.g., limiting the dataset to 100 posts per user). Our implementation does provide some commonly precomputed resources for geoinference, such as the social network or user's home locations. However, individual models may have custom needs. One potential solution is for a host to support individual workspaces for each experimenter, which persist data and results between trials of an experimental model. Such workspaces could allow also models to resume from a partially-finished state which may aid in debugging and avoid recomputing initial results when testing multiple variations of a single model. However, workspaces introduce additional security concerns for maintaining data privacy.

Development Ease Customarily, most computational methods are iteratively designed, with extensive debugging and manually analyzing problematic cases. This development process becomes more complicated for a FREESR-hosted task because access to the dataset is restricted. The current implementation mitigates this complication to some degree by (1) providing error tracebacks through the the Job Dashboard if a method encounters problems when running and (2) if the job runs successfully, returning a random sample of 10,000 tweet IDs and the location predictions made by those methods. The first feature allows experimenters to debug their method implementations, while the second allows inspecting the errors made by the model.

Nevertheless, we anticipate that for many experimenters working with FREESR-based tasks, the development process will pose a significant challenge but one which operators may address in three ways. First, the operator must ensure that sufficient information for debugging and improvement is returned, and when possible, allow for experimenters to test and develop their approaches locally using a subset of the data. Second, the scale of the data may not have been previously encountered by experimenters, leading to unexpected performance difficulties. Hence, an operator may provide the ability for the experimenter to create arbitrary amounts of artificial data locally to aid them in testing the scalability of their method. Third, because much or all of the sensitive data is potentially never seen by an experimenter, an operator must make sure that the characteristics of the data are thoroughly specified, e.g., for text data, specifying in which languages the text is written.

Cost and Longevity A research task can attract significant amounts of interest, which may span several years depending on the scale and difficulty of the problem. For example, the movie recommendation competition for the Netflix Prize attracted over 44,000 submissions from more than 5,000 teams over a three-year period. If the task is FREESR-based, this community interest potentially raises issues for an operator, who must provide the resources required to

[9] We acknowledge that providing some output back to the experimenter creates a limited opportunity for establishing a side channel for communicating the sensitive data, e.g., encoding post content using a binary representation in the location predictions. However, this risk is extremely limited since (a) the experimenter has no control over which posts are sampled and (b) the bandwidth of such a side channel would be limited by the experiment's ability to repeatedly run new experiments.

host and maintain the task over long periods of time. The costs required for maintaining a task may also create a negative incentive for new researchers or those in emerging regions without access to significant funding to sustain new FREESR-based tasks. As a potential solution, we are testing the feasibility of hosting tasks with cloud-based services such as DigitalOcean, which could provide a way for experimenters to fund the cost of their own experiments while still maintaining secured API-only access to the sensitive data. However, significant engineering and hosting challenges remain for this issue to be resolved effectively.

A related issue arises once an operator can no longer maintain a host. The non-transferable nature of the data potentially prohibits other operators from assuming responsibility for hosting the task. While the results themselves may be persisted on websites within the research community, additional solutions will need to be explored for transitioning datasets between hosts. One exciting possibility would be that the data owners themselves host FREESR-based tasks for the research problems in which they are interested; such industry-sponsored tasks could offer new insights into research problems using previously-inaccessible data.

Unaddressed Challenges While our geoinference task was largely successful at addressing the challenges posed by the FREESR paradigm, two remain unaddressed: Challenge 4 for protecting intellectual property and the related Challenge 6 for supporting proprietary resources. Challenge 4 requires that the host operator is prevented from examining submission content, effectively treating it as a black box. To address this challenge, we initially tested having the submission procedure automatically use code obfuscation on the software prior to transmission, which would render the code uninterpretable to the operator. However, while obfuscation libraries are available for both of the supported languages, obfuscating the submission proved extremely difficult because the experimenter's software must conform to the software-based API; the changes to the class and method names resulted in the submission software not correctly implementing the API contract and being rejected by the validation step. Addressing the legal issues surrounding intellectual property for Challenge 6 will likely require more than purely technical changes in order to allow for proprietary software to be used on the host, but is nonetheless essential for allowing all types of experimenters to participate.

Insight Beyond developing computational methods, research on a shared task often provides significant insight into which parts of the task are most difficult. For example, prior geoinference work has noted the difficulty of inferring locations for individuals that are active travelers across large distances (Compton, Jurgens, and Allen 2014). Our FREESR-based task still enables researchers to provide this type of insight because IDs for samples of the data are returned, thereby allowing the data to be reconstructed and analyzed by the experimenter. However, in some settings, the data may not be reconstructable on the experimenter's side at all, such as for medical records, sensitive images, or for social media platforms without an API. While computational methods can still be developed in these settings, the lack of data access can potentially stymie meaningful insight into the task's particular research problem.

8 Conclusion

Social media presents two significant obstacles for comparative evaluation due to limitations on sharing datasets and the temporal nature of the datasets themselves, where content may be deleted over time. This paper has presented a new evaluation paradigm, FREESR, designed to address both challenges. To allow access to sensitive data, a FREESR-based task allows experimenters to submit their methods for execution within a secure, remote environment that provides programmatic access to the dataset without the possibility of data redistribution. Using a case study on a real-world problem, we meet a current research community need by implementing and hosting a FREESR-based geoinference task, showcasing the benefits of the model and identifying how many of the challenges can be addressed. Both the task and the hosting software are made available to the research community for use at `http://networkdynamics.org/resources/software/`. Beyond the implementation, we see the FREESR paradigm as being an important step towards making study reproducibility and method comparison more principled and ubiquitous in the social media research community, and potentially expanding the paradigm to other research areas using sensitive, non-transferable data such as those with medical records.

References

Almuhimedi, H.; Wilson, S.; Liu, B.; Sadeh, N.; and Acquisti, A. 2013. Tweets are forever: a large-scale quantitative analysis of deleted tweets. In *Proceedings of CSCW*, 897–908. ACM.

Braschler, M., and Peters, C. 2004. Cross-language evaluation forum: Objectives, results, achievements. *Information Retrieval* 7:7–31.

Compton, R.; Jurgens, D.; and Allen, D. 2014. Geotagging one hundred million twitter accounts with total variation minimization. In *IEEE International Conference on Big Data*.

Eisenstein, J.; O'Connor, B.; Smith, N. A.; and Xing, E. P. 2010. A latent variable model for geographic lexical variation. In *Proceedings of EMNLP*, 1277–1287.

Graham, M.; Hale, S. A.; and Gaffney, D. 2014. Where in the world are you? geolocation and language identification in twitter. *The Professional Geographer* 66(4):568–578.

Grishman, R., and Sundheim, B. 1996. Message understanding conference-6: A brief history. In *Proceedings of COLING*, volume 96, 466–471.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. H. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter* 11(1):10–18.

Hecht, B.; Hong, L.; Suh, B.; and Chi, E. 2011. Tweets from justin bieber's heart: the dynamics of the location field in user profiles. In *Proceedings of CHI*, 237–246. ACM.

Jurgens, D.; Finethy, T.; McCorriston, J.; Xu, Y. T.; and Ruths, D. 2015. Geolocation prediction in twitter using social networks: A critical analysis and review of current practice. In *Proceedings of the 9th International AAAI Conference on Weblogs and Social Media (ICWSM)*.

Jurgens, D. 2013. That's what friends are for: Inferring location in online social media platforms based on social relationships. In *Proceedings of ICWSM*.

Kilgarriff, A., and Palmer, M. 2000. Introduction to the special issue on senseval. *Computers and the Humanities* 34(1-2):1–13.

Li, R.; Wang, S.; Deng, H.; Wang, R.; and Chang, K. C.-C. 2012. Towards social user profiling: unified and discriminative influence model for inferring home locations. In *Proceedings of KDD*, 1023–1031. ACM.

Li, R.; Wang, S.; and Chang, K. C.-C. 2012. Multiple location profiling for users and relationships from social network and content. *Proceedings of the VLDB Endowment* 5(11):1603–1614.

Lingad, J.; Karimi, S.; and Yin, J. 2013. Location extraction from disaster-related microblogs. In *Proceedings of WWW*, 1017–1020.

Liu, Y.; Kliman-Silver, C.; and Mislove, A. 2014. The Tweets They are a-Changin': Evolution of Twitter Users and Behavior. In *Proceedings of ICWSM*.

Mahmud, J.; Nichols, J.; and Drews, C. 2012. Where is this tweet from? inferring home locations of twitter users. In *Proceedings of ICWSM*.

McCreadie, R.; Soboroff, I.; Lin, J.; Macdonald, C.; Ounis, I.; and McCullough, D. 2012. On building a reusable twitter corpus. In *Proceedings of SIGIR*, 1113–1114. ACM.

McMinn, A. J.; Moshfeghi, Y.; and Jose, J. M. 2013. Building a large-scale corpus for evaluating event detection on twitter. In *Proceedings of CIKM*, 409–418. ACM.

Morante, R., and Blanco, E. 2012. *SEM 2012 Shared Task: Resolving the scope and focus of negation. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM)*, 265–274. Association for Computational Linguistics.

Morstatter, F.; Pfeffer, J.; Liu, H.; and Carley, K. M. 2013. Is the Sample Good Enough? Comparing Data from Twitter's Streaming API with Twitter's Firehose. In *Proceedings of ICWSM*.

Nakov, P.; Kozareva, Z.; Ritter, A.; Rosenthal, S.; Stoyanov, V.; and Wilson, T. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Proceedings of SemEval*.

Ounis, I.; Macdonald, C.; Lin, J.; and Soboroff, I. 2011. Overview of the TREC-2011 microblog track. In *Proceedings of the 20th Text REtrieval Conference (TREC 2011)*.

Pak, A., and Paroubek, P. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of LREC*, volume 10, 1320–1326.

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

Petrovic, S.; Osborne, M.; and Lavrenko, V. 2010. The Edinburgh Twitter Corpus. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media*, 25–26.

Rosenthal, S.; Nakov, P.; Ritter, A.; and Stoyanov, V. 2014. Semeval-2014 task 9: Sentiment analysis in twitter. In *Proceedings of SemEval*.

Rout, D.; Bontcheva, K.; Preoţiuc-Pietro, D.; and Cohn, T. 2013. Where's @Wally?: a Classification Approach to Geolocating Users Based on Their Social Ties. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media*.

Schwartz, H. A.; Eichstaedt, J. C.; Kern, M. L.; Dziurzynski, L.; et al. 2013. Characterizing geographic variation in well-being using tweets. In *Proceedings of ICWSM*.

Strnadova, V.; Jurgens, D.; and Lu, T.-C. 2013. Characterizing online discussions in microblogs using network analysis. In *Proceedings of the AAAI Spring Symposium on Analyzing Microtext*.

Tjong Kim Sang, E. F., and Buchholz, S. 2000. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proceedings of CoNLL*, 127–132. Association for Computational Linguistics.

Voorhees, E. M.; Harman, D. K.; et al. 2005. *TREC: Experiment and evaluation in information retrieval*, volume 1. MIT press Cambridge.