PREDICTIVE VISUAL ANALYTICS OF SOCIAL MEDIA DATA FOR SUPPORTING REAL-TIME SITUATIONAL AWARENESS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Luke S. Snyder

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Computer Science

May 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF THESIS APPROVAL

Dr. David S. Ebert, Chair

School of Electrical and Computer Engineering

Dr. Dan Goldwasser

Department of Computer Science

Dr. Jeremiah Blocki

Department of Computer Science

Approved by:

Dr. Christopher Bingham

Head of the Department Graduate Program

To my parents.

ACKNOWLEDGMENTS

First and foremost, I would like to thank Dr. Ebert for his invaluable guidance, support, and encouragement over the past two years. I would also like to thank Dr. Morteza Karimzadeh for his endless support and wisdom. Finally, a sincere thank you to my colleagues at the VACCINE Center — Yi-Shan Lin, Jieqiong Zhao, Guizhen Wang, Calvin Yau, Dr. Audrey Reinert, Dr. Jingjing Guo, and Christina Stober — for their assistance and advice.

Next, I would like to thank my parents, who have loved and sacrificed a great deal for me throughout my life. Words cannot express my love and gratitude for you. I would also like to thank my siblings — Jake, John, and Hope — for their constant encouragement. I love you all dearly.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Snyder, Luke S. M.S., Purdue University, May 2020. Predictive Visual Analytics of Social Media Data for Supporting Real-time Situational Awareness. Major Professor: David S. Ebert.

Real-time social media data can provide useful information on evolving events and situations. In addition, various domain users are increasingly leveraging real-time social media data to gain rapid situational awareness. Informed by discussions with first responders and government officials, we focus on two major barriers limiting the widespread adoption of social media for situational awareness: the lack of geotagged data and the deluge of irrelevant information during events. Geotags are naturally useful, as they indicate the location of origin and provide geographic context. Only a small portion of social media is geotagged, however, limiting its practical use for situational awareness. The deluge of irrelevant data provides equal difficulties, impeding the effective identification of semantically relevant information. Existing methods for short text relevance classification fail to incorporate users' knowledge into the classification process. Therefore, classifiers cannot be interactively retrained for specific events or user-dependent needs in real-time, limiting situational awareness. In this work, we first adapt, improve, and evaluate a state-of-the-art deep learning model for city-level geolocation prediction, and integrate it with a visual analytics system tailored for real-time situational awareness. We then present a novel interactive learning framework in which users rapidly identify relevant data by iteratively correcting the relevance classification of tweets in real-time. We integrate our framework with the extended Social Media Analytics and Reporting Toolkit (SMART) 2.0 system, allowing the use of our interactive learning framework within a visual analytics system adapted for real-time situational awareness.

1 INTRODUCTION

Social media data has been used extensively in a variety of applications and research endeavors due to its ability to provide useful information on the public's opinions and behavior. This can be especially useful in assessing and understanding various situations. For instance, first responders are leveraging Twitter data to obtain actionable information for crisis response and prevention (see [1] for an extensive list of literature on this subject), such as identifying people in need of help during a natural or human-caused disaster event.

However, our discussions with emergency responders and government officials revealed several obstacles currently facing the effective use of social media data for situational awareness [2]. In this work, we focus on two of the most frequently cited issues: (1) the lack of sufficient geotagged data and (2) the deluge of irrelevant information, impeding social media analysts' ability to find useful and important data.

In this chapter, we characterize both issues and propose predictive visual analytics solutions in sections 1.1 and 1.2, respectively. We then present our thesis statement and contributions in section 1.3 and describe this work's outline in section 1.4.

1.1 Geolocation Inference of Social Media Data

Geotagged tweets — tweets containing geographic coordinates from the issuing device — are important for situational awareness as they provide the location of posting. Without location context, first responders are unable to decide where or how to respond to information they receive. This is especially true during natural disasters when geographic context is necessary for dispatching appropriate emergency response. Furthermore, tweets' originating locations are important in other domains such as sentiment analysis or digital marketing. However, tweets are not geotagged by default,

requiring Twitter users to manually activate geotagging [3,4]. As a result, only 0.9% of all tweets are geotagged [3,4], considerably limiting their use for situational awareness.

To increase the amount of geotagged tweets, researchers have developed various algorithms for predicting the location of a tweet, such as deep learning classifiers [5–8] and gazetteer-based methods [9]. However, Twitter's public feed has undergone changes that may affect such algorithms. Further, the utilization of state-of-the-art prediction models in real-time visual analytics systems has not been explored. As a result, we (i) adapt an existing geolocation prediction method, computationally improve its accuracy, and integrate it with SMART [10, 11], a visual analytics system designed to facilitate situational awareness, (ii) demonstrate the utility of geolocation prediction for real-time systems, and (iii) evaluate the effectiveness of the integrated system using Twitter's public feed in September 2019.

1.2 Interactive Machine Learning of Social Media Data

Analysts in various domains are increasingly using social media to gain rapid situational awareness. However, the vast amounts of unstructured text make the identification of relevant information difficult, thereby limiting situational awareness. This issue is further compounded by changes in topics of interest (to end users) over time, since the computational models built to determine relevant information for one event or one user group may not apply to other events or other user groups due to variations in diction, word structure, or user expectations.

Several classification approaches have been developed to identify relevant and irrelevant social media information, such as clustering [12, 13], keyword matching [14], and term-vector similarity [15]. However, at the time of writing, no existing work in this area includes interactive learning and retraining with real-time data, focusing instead on improving the machine learning algorithms themselves [13, 14, 16–23] or interactively training on archived datasets [24, 25]. In addition, while active learning systems can provide efficient interactive training [26], users are unable to relabel in-

correct classifications and retrain the model to improve performance. Continuing on our example of first responders, a pre-trained classifier may not fulfill first responders' varying needs, since one first responder may be interested in monitoring road closures, and another one might be interested in identifying disinformation and misinformation on social media in order to take counter-action. Ultimately, first responders' definition of relevancy will depend on the situation at hand, which can vary over time. Interactively training classifiers through iterative user labeling can alleviate this problem.

In this work, we present a novel interactive framework in which the user iteratively (re)labels the relevancy of streaming social media data to adaptively train the underlying model to match their needs for improved situational awareness. We compare three different types of neural networks in terms of classification performance and computational efficiency for real-time learning. Furthermore, we optimize and computationally evaluate the selected models by simulating the real-time user feedback on several crisis-related datasets. Our results show that our interactive model outperforms state-of-the-art machine learning-based classification models.

To incorporate our evaluated models into a working application, we extend an existing visual analytics system tailored for situational awareness called the Social Media Analytics and Reporting Toolkit (SMART) [11, 27], which has been successfully used by many first responder groups in the United States. SMART allows users to interactively explore trending topics on social media through integrated topic modeling and spatial, temporal, and textual visualizations. We call the newly extended system SMART 2.0, which incorporates our interactive learning framework to address the needs raised by the aforementioned first responder users and reduce noise in the incoming stream of data.

Finally, we present domain-expert feedback on the usefulness of our approach as experienced by multiple first responders who used SMART 2.0 for crisis-related use cases. In addition, we include two usage scenarios of the system to illustrate its application to real-life situations.

1.3 Thesis Statement

The thesis statement of this work is as follows:
Geolocation inference and interactive machine learning grounded in visual analytics enables social media analysts to effectively identify relevant information and utilize geographic context for improved situational awareness.

The major contributions of this paper can be classified into two categories. The first is a geolocation inference approach for real-time visual analytics that increases the amount of geotagged data, thereby providing increased situational awareness for first responders and social media analysts. The second is a coupled interactive learning framework and visual analytics system that enables various domain users to effectively identify relevant tweets and adapt classifiers to their needs during real-time situations. Overall, the contributions can be summarized as follows:

1. Geolocation inference approach for visual analytics:
    (a) We adapt and computationally improve an existing geolocation prediction method for social media data [7].
    (b) We integrate the improved geolocation prediction method with a visual analytics system and evaluate its effectiveness using Twitter's public feed in September 2019.
    (c) We demonstrate the utility of geolocation prediction for real-time systems.
2. Interactive learning framework and visual analytics system:
    (a) We present a novel interactive learning framework for classification of streaming text data that allows users to (re)train models for improved performance.
    (b) We compare three different types of neural networks in terms of performance and computational efficiency, and tune the models for learning at interactive rates. We further computationally evaluate the selected model on several disaster-related datasets.

(c) We integrate our models in SMART 2.0, a visual analytics application for situational awareness, and present user feedback obtained from domain experts using the system for crisis events.

1.4 Outline

In the remainder of this work, we discuss the background and related work in chapter 2, our geolocation inference approach for visual analytics in section 3, our interactive learning framework for improving social media relevance classification in chapter 4, and concluding remarks and future work in chapter 5.

2 RELATED WORK

In this chapter, we first review the background and related work for geolocation inference in section 2.1. We then review the background and related work for short text classification and visual analytics-supported interactive learning in sections 2.2 and 2.3, respectively.

2.1 Geolocation Inference

Researchers have developed various techniques to estimate geographic locations of both Twitter users and tweets themselves. In general, there are three primary levels of tweet geolocation prediction: the event level, user level, and tweet level.

Geolocation inference at the event level estimates the location of events mentioned in text. This level of inference predominantly relies on geoparsing — the process of identifying geolocations in text and disambiguating between multiple toponym references — and has been studied extensively [28–30]. Recent studies integrated Twitter metadata and named entity recognition algorithms into geoparsing approaches and obtained high accuracy percentages over 90% [3, 29]. However, event geolocation inference might not reflect the actual location of individual tweets.

Geolocation inference at the user level estimates the location of Twitter users based on their tweet history and other useful information. Specifically, user locations can be predicted by utilizing toponym references within their tweets as well as user metadata such as friend networks and time zones [5]. The majority of techniques for user-level prediction utilize either state-of-the-art statistical models or machine learning [31]. Qian et al. [32] designed a probabilistic machine learning graph model, obtaining high accuracy for predicting users' geolocations at the country or state

level. Do et al. [5] trained a multi-entry neural network to predict users' locations, yielding an accuracy of over 60%.

Geolocation inference at the tweet level estimates the location of individual tweets. This differs from user-level prediction in that a tweet might be posted in a separate location from where they live, such as during a vacation or work hours. In general, tweet-level prediction, which is crucial for situational awareness, is a difficult task. Tweet content might contain toponyms that do not reflect the tweet's actual origin, and tweets may not contain sufficient metadata (e.g., time zone) or useful content for prediction. However, researchers have continued to explore potential solutions and machine learning model advancements. Thom et al. [33] provided probabilistic methods that analyze term density spatial patterns and user movement histories from large datasets to predict locations. Duong-Trung et al. [34] developed near real-time geolocation prediction at the tweet level with a matrix factorization-based statistical regression model. Li et al. [8] adapted a Bayesian model and a convolutional long short-term memory (LSTM) neural network to construct a user location history and predict individual locations of future tweets. Lau et al. [7] designed deepgeo, an end-to-end neural network that combines various recurrent and convolutional neural networks for inferring tweet locations at the city level, achieving a state-of-the-art accuracy of approximately 40%. To facilitate the assessment and comparison of different location inference methods, Mahtal et. al [35] provided a comprehensive analytic workbench with prediction and error visualizations.

While considerable attention has been devoted to advancing state-of-the-art models for geolocation prediction, their use, adaptation with updated Twitter feed, and evaluation within a real-time visual analytics system have not been explored, which is the focus of our work.

2.2 Short Text Classification

Researchers have presented many techniques to classify text documents into categories such as sentiment or topics [36–39]. However, classifying short text, e.g. social media posts, is more challenging due to the lack of contextual information and loose adherence to standard grammar. To tackle the brevity of short text, auxiliary resources such as external corpora [40] or knowledge bases [41], or methods such as term frequency-inverse document frequency (TF-IDF) [16], have been proposed for improving classification.

Representing words as n-dimensional vectors (i.e. word embedding) has become increasingly prevalent, since vectors can be used as inputs to machine learning models for finding semantic similarities [42, 43]. In particular, Google's `Word2Vec` [44] has been employed extensively in classification tasks [19, 20, 45–47] due to its impressive ability in capturing linguistic regularities and semantics. For instance, words frequently used together are likely to be closer in the `Word2vec` vector space than words that are not, and vector operations reveal meaningful semantics (e.g., the vector "King" − "Man" + "Woman" is close to the vector "Queen" [44]). Since pre-trained `Word2vec` models encode embeddings learned from larger web corpora, they have been increasingly used in short text classification tasks [19, 20, 47, 48].

Neural networks have generated state-of-the-art results in recent years for text classification problems [19, 20, 44, 49] and have also been used with `Word2Vec` [19, 20, 48]. Neural networks are well-suited for online learning processes in which training data is supplied iteratively since they can learn adaptively from new data [19, 20]. Nguyen et al. [20] presented a convolutional neural network with `Word2Vec` that outperformed non-neural classifiers, and Nguyen et al. [19] proposed a new online learning classification algorithm for deep neural networks utilizing the log-loss and gradient of sequential training batches. Their methods were evaluated with disaster-related datasets. However, these methods were not adapted to user-guided learning in which time constraints are essential and the provided batches may be small. In particular,

the online learning method designed by Nguyen et al. [19] was evaluated with batch sizes of 200. In our work, we assume the user needs to train with flexibly interactive amounts of data (10–20 samples) to view immediate predictive improvements for situational awareness.

Classification for situational awareness. Utilizing real-time social media data for situational awareness (and crisis prevention in particular) is a heavily researched topic [14, 16–21, 23]. However, identifying situationally-relevant information is non-trivial due to the high noise-to-signal ratio. Karimi et al. [17] found that classification methods, such as Support Vector Machine and multinomial Naïve Bayes, can identify disaster-related tweets, although generic features such as hashtag count and tweet length are preferable so that the model does not learn relevancy only for a specific disaster. Researchers have used clustering [12, 13, 22] or enhanced keyword matching [14] to detect relevant crisis and event information, and provided human-annotated Twitter corpora that can be used to train word embedding models [50].

Nazer et al. [18] developed a system to detect requests for help by utilizing both tweet context (e.g., geotag) and content (e.g., URLs). Rudra et al. [21] designed a novel classification-summarization framework to classify disaster-related tweets, and then summarize the tweets by exploiting linguistic properties typical of disaster tweets (e.g., combinations of situational and non-situational information). Zoppi et al. [23] provided a relevance labeling strategy for crisis management that computed data relevance as a function of the data's integrity (e.g., are the geo-coordinates incorrect?), statistical properties (e.g., can we select a subset of the data that are geographically close?), and clustering (e.g., what groups are present in the data?). Toriumi et al. [22] clustered tweets based on their retweet count in real-time to extract important topics and classify tweets accordingly.

The methods discussed so far, however, lack user interactivity. In particular, these classification methods are inflexible to user-dependent needs that change over time as

new situations and events occur. As such, their practical use for real-time situational awareness is limited.

2.3 Visual Analytics and Interactive Learning for Situational Awareness

Researchers have presented a number of visual analytics (VA) solutions for situational awareness. Diakopoulos et al. [15] developed Vox Civitas, a VA application for journalistic analysis and user-guided filtering using social media content. Vox Civitas filters out unrelated data by automatically computing time-dependent term-vector similarities. TwitInfo [51] aggregates streamed Twitter data and automatically discovers events from activity peaks in real-time. The authors assign relevance to a tweet by counting its number of event-related keywords. Pezanowski et al. [52] designed the geovisual analytics system SensePlace3 to provide situational awareness by leveraging geographical information and place-time-theme indexing with string-based queries for exploring datasets. SensePlace3 primarily relies on TF-IDF for tweet retrieval in response to user queries. However, these tools do not employ machine learning for relevance classification and do not integrate user feedback to improve their underlying models or algorithms.

Visual analytics has also been increasingly used to improve various machine learning processes, such as feature selection [53], attribute weighting [54], and labeling [24, 25, 55], and even understanding the models themselves [56–58]. Sacha et al. [59] proposed a framework to discuss the various forms of human interaction with machine learning models in visual analytics systems and theorized that VA tools could increase knowledge and usability of machine learning components. Endert et al. [60] designed a system that classifies archived documents through user-guided semantic interactions (e.g., moving a document to another group) that improve the underlying model. Our work is based on the same idea in that we intend to improve model performance through user feedback, but with real-time social media data.

Heimerl et al. [25] analyzed three separate methods for user-guided classification of a set of archived text documents: the *basic* method, which does not employ sophisticated visuals; the *visual* method, which visually represents the labeled and unlabeled documents for user exploration; and the *user-driven* method, which provides the user with full control over the labeling process. The first two methods employ active learning, in which the model selects a data sample to be labeled by the user that most effectively helps it distinguish relevant from irrelevant data. This contrasts with the user deciding which instances they wish to label. The authors did not find any statistically significant differences in terms of F_1 score between the methods in their user study. Bosch et al. [24] developed ScatterBlogs2, a VA application that provides user-guided learning of filter classifiers on historical social media messages to support situational awareness. These two works are perhaps the most similar to ours, yet differ in two fundamental ways. First, they do not provide interactive learning in real-time, which strains the user, as they are required to visit historical data for additional training. Second, they do not employ neural networks, which are better suited for online learning environments, such as social media streaming, in which training data is supplied sequentially over time [19, 20]. It is important to note that Bosch et al. [24] allow the user to adjust a filter's focus (i.e., how precise the classification is) in real-time if it misses relevant data or does not sufficiently filter out irrelevant data. However, this could indicate that the model has not properly learned the distinction between relevant and irrelevant data. Since training can only be completed with historical posts, the user is unable to update the model immediately with the streamed data, limiting situational awareness. Our approach not only solves this issue by allowing the user to immediately train the model for improvement, but also provides the user with the ability to create classifiers on-the-fly to accommodate their real-time needs.

Active Learning Active learning, as briefly discussed in the previous paragraph, is a machine learning method that queries an oracle (e.g., human annotator) to provide

labels for selected instances [61]. The active learner can select instances that are, for instance, closest to the model's decision boundary [62], or have the lowest probability for their class [63]. While traditional supervised learning requires sufficient amounts of labeled data, active learning does not. Rather, the active learner may begin with a small set of labeled data and query instances that are maximally informative, therefore achieving high performance with minimal labeling [26].

Active learning techniques have been widely employed in various learning tasks, such as social media stance classification [64] and post-crisis relevance filtering [65]. Bosetti et al. [66] provided an active learning platform for efficient labeling of thousands of microblogs. The authors also developed a novel query procedure that selects messages with top bigrams and retweet counts for labeling. Hu et al. [67] also contributed a novel active learning selection mechanism for social media data by exploiting social network relationships.

Our interactive learning framework differs from active learning in two primary respects. First, active learning lacks transparency, as users may be unaware of the label querying procedure. We provide the user full control over labeling to maintain transparency. Second, users are unable to retrain models in active learning environments since they are only queried for selected instances without labels. Our framework notably differs in this respect, as we allow users to relabel all incorrectly classified data to retrain the model for immediate improvement.

3 CITY-LEVEL GEOLOCATION OF TWEETS FOR REAL-TIME VISUAL ANALYTICS

This chapter is based on the paper published in SIGSPATIAL GeoAI 2019:

Luke S. Snyder, Morteza Karimzadeh, Ray Chen, and David S. Ebert. City-level geolocation of tweets for real-time visual analytics. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery (GeoAI)*, 2019

In this chapter, we present and discuss our geolocation inference approach for visual analytics to increase the amount of geotagged data and improve situational awareness. We adapt and improve an existing deep learning geolocation prediction model called deepgeo [7], and then integrate the improved model with a visual analytics system. We evaluate both the improved model's accuracy and the visual analytics system's effectiveness.

3.1 Deepgeo

Our visual analytics system (discussed in Section 3.2.2) leverages real-time streaming tweets to facilitate situational awareness for first responders. As such, the location of individual tweets is paramount for assessing the situation and responding appropriately. Of the various tweet-level geolocation prediction models, deepgeo [7] is the most recent state-of-the-art, open-source model. Being open source was important since we wished to build on previous research and adapt, improve, and integrate a well-tested model into our visual analytics application. In this section, we discuss our adaptations and improvements to deepgeo.

3.1.1 Overview and Adaptations

Deepgeo is a deep learning model that predicts the geographic locations of individual tweets at the city level. Deepgeo's original model takes six feature inputs: (1) tweet text; (2) tweet creation time; (3) user UTC offset; (4) user time zone; (5) user-listed profile location (text); and (6) user account creation time. Each of the six features are individually processed by distinct neural networks and concatenated before the final prediction layer. The output of the model is one of the 3,362 possible city labels from the training data, each represented as an integer (from 0 to 3,361).

Since deepgeo's release in 2017, Twitter has made important changes to the user metadata. In particular, the user time zone and user UTC offset are no longer provided, negatively impacting the accuracy. To adapt to these restrictions, we removed the time zone and UTC offset features from deepgeo, leaving the remaining four: tweet text, tweet creation time, user location, and user account creation time.

3.1.2 Improvement using Word2Vec Embeddings

Deepgeo processes the tweet textual content with a character-level recurrent convolutional network with max-over-time pooling and self-attention. However, the character embeddings are initialized with a random uniform distribution and learned with subsequent training. This may negatively impact performance since the embedding weights are initially not learned and therefore not meaningful.

As an alternative choice for embedding weights, we used Google's skip-gram `Word2Vec` [44, 69] in our improved model, which we call deepgeo2. `Word2Vec` is a pre-trained (i.e., initially learned) model that provides embedding weights at the word level. `Word2Vec` contains 3 million 300-dimensional word vectors pre-trained on the Google News corpus with 100 billion words. Various machine and deep learning algorithms have utilized `Word2Vec` and achieved state-of-the-art results in text classification, primarily because `Word2Vec` embeddings strongly capture semantic and

syntactic relationships between words (e.g., the vector "King" - "Man" + "Woman" is close to the vector "Queen" [44]).

As with the original character embeddings, each token (word) is sequentially represented with its vector embedding (`Word2Vec`) and concatenated with the forward and backward hidden states from a bi-directional LSTM network before applying max-over-time pooling, self-attention, and weighted mean (the existing architecture of deepgeo's text network). Also, as with deepgeo, we did not preprocess or clean the tweet text, which we will investigate in the future. If a word's 300-dimensional vector is not present in the `Word2Vec` pre-trained model, we randomly initialize it [70].

3.2 Evaluation

In this section, we evaluate the accuracy improvement of deepgeo2 (utilizing `Word2Vec`) compared to the original character embeddings, and present and discuss our visual analytics system that utilizes deepgeo2 to predict the location of real-time tweets to facilitate situational awareness. We further evaluate and demonstrate the usefulness of the added predictive functionality.

3.2.1 Geolocation Prediction

To evaluate the effectiveness of our improvement using `Word2Vec` embeddings, we trained our adaptation of deepgeo twice: once with the original character-level embeddings and once with `Word2Vec` embeddings (deepgeo2). The training and testing processes were executed in the same manner and with the same optimized hyperparameters that Lau et al. [7] originally used. The only difference was with the amount of training data. Lau et al. trained with 9.8 million tweets from a geolocation prediction shared task dataset[1]. However, due to Twitter terms of service, the dataset only provides the tweet IDs, requiring the developers to manually download the tweets and metadata associated with each tweet ID, which could take up to 30–40 days due to

[1] `https://noisy-text.github.io/2016/geo-shared-task.html`

download rate limits. As such, due to time constraints for reporting on our ongoing research, we trained with the first-downloaded 350,000 tweets.

As shown in Table 3.1, the original deepgeo model achieved a precision of 0.38, recall of 0.32, F_1 score of 0.35, and accuracy of 31.6%, while our improved deepgeo2 yielded an increased 0.39 precision, 0.34 recall, 0.36 F_1 score, and 34.2% accuracy, which is considerable in tweet geolocation prediction research. Although it is likely that the increase in these metrics may fluctuate with more training data, we expect deepgeo2 with `Word2Vec` embeddings to continue outperforming the original model.

Table 3.1.
Precision, recall, and accuracy metrics for the deepgeo and deepgeo2 models.

Model	Precision	Recall	F_1 score	Accuracy
Deepgeo	0.38	0.32	0.35	31.6%
Deepgeo2	0.39	0.34	0.36	34.2%

3.2.2 Integration with SMART

The Social Media Analytics and Reporting Toolkit (SMART) [10, 11] is a system for visual analysis of geotagged, publicly-available real-time tweets to enhance situational awareness and expedite emergency response. SMART has been used by over 300 first responders in 70 organizations for major events, such as presidential inaugurations and sports games. Such users require as much data as possible for effective analysis and better coverage. SMART provides several integrated visualizations for interactive exploration and anomaly detection, such as topic-modeling, spatial clustering, and temporal views (Figure 3.1).

SMART already streamed and visualized geotagged tweets. To incorporate geolocation inference into SMART, SMART first collects real-time tweets with user-listed profile locations (as strings, such as “Lafayette, Colorado”), but no geotag, (since

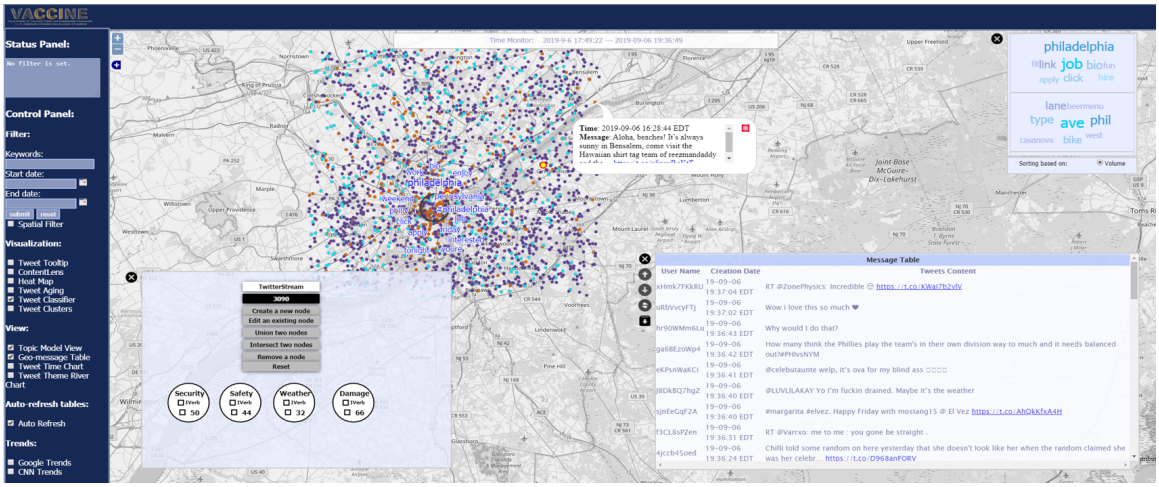

Figure 3.1. SMART allows users to interactively explore and identify tweets through tools such as topic modeling, spatial filtering, and temporal visualization. Geotagged tweets are colored purple, Instagram posts are colored orange, and predicted tweets are colored light blue.

the user profile is one of the four required model inputs) at a rate of about 400–700 tweets per minute within the entire United States. This is relatively low since SMART uses the free, rate-limited Twitter streaming API[2]. After 512 tweets are collected (the model's batch size), which occurs approximately every minute, they are transmitted to deepgeo2 for location prediction and then visualized. Currently, tweets with inferred locations are placed in random locations within the bounds of the geolocated city to keep SMART's other aggregate-based visualizations (including the spatial topic modeling and word clouds) consistent. However, tweets with predicted locations are symbolized using a different color (Figure 3.1) from explicitly geotagged data to indicate to the user that their locations are estimated and not exact. As part of our future research, we will improve our cartographic representation of city-level estimated location of predicted locations.

[2]`https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets`

To ascertain the geolocation prediction model's effect on SMART's utility (i.e., how much more data SMART could collect for user analysis), we assumed the role of a SMART user observing streamed tweets in real-time. We analyzed three cities — Philadelphia, PA; Chicago, IL; and New York, NY, east of I-95 (Table 3.2) — on a desktop computer with 32 GB RAM and 2 6-core Intel(R) Xeon(R) E5-2630 CPUs at 2.30GHz. After 100 minutes of SMART use with each city (i.e., we used SMART to view tweets only located within the specified city), we measured the number of tweets with an estimated geolocation. Table 3.2 provides the results for each city: there were 475 predicted tweets in Philadelphia, PA; 1,215 in Chicago, IL; and 2,384 in New York, NY. Further, in each respective city, the geolocated tweet percentage increase was 34.30%, 48.16%, and 39.89%. As with deepgeo, deepgeo2 takes less than 2 seconds to predict the city labels of 512 tweets using the aforementioned hardware.

Our results indicate that the geolocation prediction functionality significantly improved the amount of data collected and visualized by SMART, allowing users to view and analyze more data for situational awareness. It is important to note that although more data is collected, a large portion of it might still not be accurate or relevant, which is typical of social media due to limited context. However, SMART markedly distinguishes tweets with predicted locations to inform users. In addition, SMART users have frequently indicated that they would prefer more data to less, even if it is

Table 3.2.
Number of streamed geotagged and predicted tweets for Philadelphia, Chicago, and New York City. We calculate the percent increase as the percent difference between (1) the total number of geotagged and predicted tweets and (2) the total number of geotagged tweets.

Region	Min Lat	Max Lat	Min Lon	Max Lon	Number of Geotagged Tweets	Number of Predicted Tweets	Percent Increase
Philadelphia, PA	39.86	40.13	-75.32	-74.93	1,385	475	34.30%
Chicago, IL	41.57	42.12	-88.15	-87.49	2,523	1,215	48.16%
New York, NY, East of I-95	40.49	40.91	-74.26	-73.70	5,976	2,384	39.89%

inaccurate, since they might be able to identify relevant tweets that would otherwise not be present without geolocation inference.

3.3 Summary

In this chapter, we adapted, improved, and evaluated deepgeo and presented deepgeo2, a deep learning model that infers individual tweets' locations at the city level. We integrated deepgeo2 with SMART, a visual analytics application that allows first responders to investigate real-time, geotagged tweets. Finally, we measured the increase in the number of geolocated tweets within SMART by analyzing its data collection after incorporating deepgeo2.

4 INTERACTIVE LEARNING FOR IDENTIFYING RELEVANT TWEETS TO SUPPORT REAL-TIME SITUATIONAL AWARENESS

This chapter is based on the paper published in IEEE VIS 2019:

Luke S. Snyder, Yi-Shan Lin, Morteza Karimzadeh, Dan Goldwasser, and David S. Ebert. Interactive learning for identifying relevant tweets to support real-time situational awareness. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):558–568, 2020

In this chapter, we present and discuss our novel interactive learning approach for identifying relevant streaming data in real-time. Our approach consists of an interactive learning framework in which users iteratively (re)label incoming streaming data to improve classifiers on-the-fly (Figure 4.1), and a visual analytics system called SMART 2.0 that realizes our framework to improve situational awareness. We evaluate the performance and computational efficiency of our classification model adapted for real-time learning. We also provide usage cases and domain expert feedback that demonstrate the effectiveness of our approach.

4.1 Interactive Learning Framework

Our framework for interactively learning relevant social media posts in real-time consists of two primary components. The first is a formalized set of design goals necessary to effectively facilitate situational awareness in real-time through user interactivity. The second is a detailed underlying model that is adapted to user-guided training with real-time streaming data. In Section 4.2, we discuss our implementation of the framework that realizes the design goals.

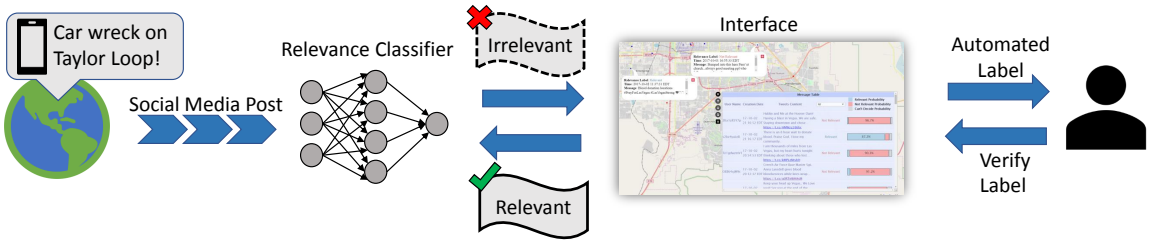

Figure 4.1. Our interactive learning framework allows users to train text relevance classifiers in real-time to improve situational awareness. In this example, a real-time tweet regarding a car accident is incorrectly classified as "Irrelevant". Through the SMART 2.0 interface, the user can view its label and correct it to "Relevant", thereby retraining and improving the classifier for incoming streaming data.

4.1.1 Design Goals

The framework's design goals were iteratively defined through discussions with domain experts such as first responders who frequently use visual analytic social media applications for real-time situational awareness. In general, these experts found it necessary for the interactive framework to incorporate user feedback in a timely manner, as well as account for time and situation-dependent user needs. With their feedback, the following specific design goals were established:

DG1 Filter and view relevant data: Filtering data by relevancy removes noisy data, allowing the user to more quickly find data that may require immediate attention or contain important information. The ability to view the relevant data itself is equally important for determining the urgency and content of relevant data.

DG2 Correct incorrect classifications: Since classifiers may provide incorrect results, especially during the early stages of training, it is necessary for the user to be able to correct the label in real-time. This both improves the model's performance and lowers the likelihood that incoming streamed data will be incorrectly classified and missed.

DG3 Create new classifiers in real-time: The needs of the user can change dramatically over time and vary across users themselves. As an example, one user may wish to train a classifier to find data related to a specific hurricane event to expedite identification of people in desperate need of assistance. However, another user may wish to find data related to safety in general, not just a hurricane. As such, they should each be able to create and train their own classifiers in real-time specific to their needs at the time.

DG4 Minimize model training time: Although it is important to design a high-performing model, time constraints are equally important. Specifically, when the model is trained by user feedback, the user should not have to wait for several minutes for the model to be retrained and relabel data. Previously streamed data labels may update with retraining, allowing the user to potentially find important information that they had not seen before. As such, it is necessary to provide these updated results as quickly as possible for real-time situational awareness.

4.1.2 Workflow

Fig. 4.2 shows the three primary components of our framework's workflow applied to streaming tweets (however, the framework can be generalized to other kinds of text). First, as tweets are streamed in real-time, they are vectorized using a word embedding model. Second, the vectorized tweets are provided as inputs to the neural network classifier (discussed in next section), which outputs a set of probabilities from the activation function of the tweet's predicted relevancy and assigns an unverified relevance label. Third, the labeled tweet is relayed to the user through the user interface. If the user identifies tweets with incorrect labels, they can correct the label for the system to retrain and improve the model for relevance predictions.

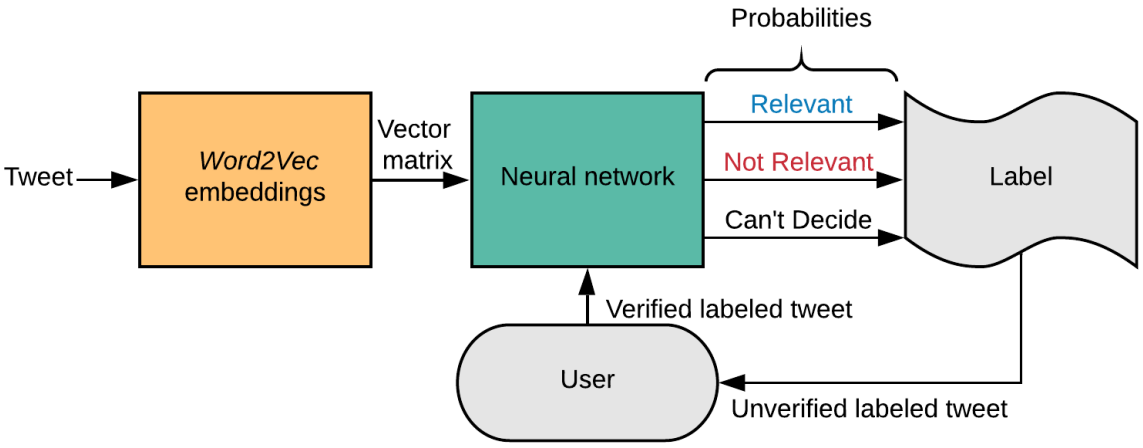

Figure 4.2. High-level workflow of our framework with three main components: tweet vectorization, tweet classification, and user feedback.

4.1.3 Interactive Model Details

In the following subsections, we elaborate on the underlying representations and models used to support our interactive learning framework. We design, optimize, and evaluate our approach with the key assumption that classifiers are trained (from scratch) in real-time using user-provided labels for streaming text. We simulate this process by adding training examples in small batches of 10 and evaluating against testing data, as explained below. All simulations were completed on a server with 128 GB RAM, 32 TB of disk storage, and 2 Intel(R) Xeon(R) E5-2640 v4 CPUs at 2.40GHz.

Model Candidates

Selecting the underlying model for our framework was a key task, as it must be efficiently trainable with a continual stream of user-labeled data (DG4). As discussed in chapter 2, neural networks are a natural choice for online learning scenarios in which training data is supplied sequentially over time [19, 20]. In addition, neural

networks have generated impressive results with `Word2Vec` [44] embeddings [19,20,48]. Therefore, we employ a neural network as our classifier to determine text relevance based on real-time training examples provided by the user. To convert the text into vector inputs (of our neural network), we use word embeddings generated by Google's `Word2Vec` skip-gram model [44, 69], which contains 3 million 300-dimensional word vectors pre-trained (and therefore, capturing word embeddings) on a subset of the Google News dataset with approximately 100 billion words.

In selecting the specific neural network model type, we experimented with the well-known Convolutional Neural Network (CNN) [71], Long Short-Term Memory (LSTM) Neural Network [72], and Recurrent Neural Network (RNN) [73] since they have performed well in various text classification tasks [74]. Hybrid architectures, such as recurrent convolutional neural networks [75], have also been proposed in recent years, but have not been made available in well-supported libraries. Therefore, we did not consider them in this work, since our goal was to also support a well-tested SMART 2.0 system for end users.

Our CNN model contains the traditional convolutional and max-pooling layers before activation [74]. Specifically, we apply a 1-dimensional convolutional layer, 1-dimensional max-pooling layer, flatten the output, and then activate it with softmax and a dense layer. The filter and kernel sizes of the convolutional layer are optimized during the hyperparameter stage (explained in Section 3.3.4). We use Hochreiter's LSTM [72] and the traditional RNN [73] architectures as provided by `Keras` [76]. The LSTM and RNN hidden layer each contain 300 hidden neurons and use softmax activation.

Design

As mentioned before, to enable the use of neural networks for classifying text, we convert the unstructured text (of the tweets) into vectors ready for consumption by the neural network. When using `Word2Vec` vectors as features for classification,

a common approach is to convert each word in the sentence to its vector, average the word vectors in the sentence, and then use the resulting feature vector for model training [47, 49]. However, averaging the vectors results in the loss of syntactic information, which can negatively impact classification results [45]. As an example, the two sentences "Only Mary will attend the ceremony." and "Mary will only attend the ceremony." would generate identical averaged sentence vectors since they contain the same set of words, but they differ in meaning. Therefore, to capture both semantic and syntactic information, we represent a sentence as a matrix where each row i is a 300-dimensional `Word2Vec` vector corresponding to word i in the original sentence.

The input to the neural network consists of the matrix representing the sentence (as described above) and the output consists of the classification labels for the input sentence (Fig. 4.2). Specifically, we allow a tweet to be (1) Relevant, (2) Not Relevant, or (3) Can't Decide. The label with the highest probability from the activation function corresponds to the final label given to it. The "Can't Decide" label indicates that the tweet may or may not be relevant depending on the context. This is useful if the user finds a social media post such as "Remembering when Hurricane Irma destroyed my home..." that may not directly relate to the current event, but may be semantically relevant, and the user does not want to mark such cases as "Not Relevant". This gives the user more flexibility to accommodate their needs since the definition of relevancy will depend on both the user and the situation.

Corpus for Model Selection and Optimization

To experiment with different neural network model types and optimize the selected model, we used a disaster-related corpus annotated on the crowd-sourcing platform, Figure Eight [77]. The dataset contains 10,876 tweets related to different types of disaster events, such as hurricanes and automobile accidents. The data was collected using keywords such as "ablaze" or "quarantine", and therefore, covers a wide variety of disaster-related topics. Our main motivation for using this open dataset is its size

(as well as topical relevance), enabling the optimization of hyperparameters and comparison of various models. In the corpus, each tweet is manually labeled by Figure Eight's workers as "Relevant", "Not Relevant", or "Can't Decide", and the distribution of labels is unbalanced. Specifically, there are 4,673 "Relevant" instances, 6,187 "Not Relevant" instances, and 16 "Can't Decide" instances. This dataset has been used in other tweet classification research projects [14]. However, the researchers of that study remove the tweets with the "Can't Decide" label to improve training data quality. As explained in the previous section, we find the "Can't Decide" option useful for users to apply to cases with insufficient context for relevance determination. We randomly shuffle the data and divide the dataset into 80% training, 10% validation, and 10% testing sets.

It is important to note that we only use the Figure Eight dataset to optimize the hyperparameters and provide an initial evaluation of the model by simulating the provision of labels in real-time by the user. Since each tweet in the dataset contains true labels that were manually assigned by humans, it allows us to evaluate the model performance by comparing the model's predictions to the true labels after each training iteration. Our proposed approach as well as its integration within the SMART 2.0 system, however, allows for the creation of the models from scratch (with no prior training) (DG3), leveraging real-time labels provided by users on streaming data for training.

Optimization

In order to experiment with the different neural network model types, we ran several training simulations with random combinations of hyperparameters (i.e., random grid search) to see which model converged to the best F_1 score. The F_1 score is a metric widely used to evaluate the quality and performance of machine learning models and neural networks [78]. It is computed as the harmonic mean of *precision* (the proportion of true positive predictions compared to the total number of positive

predictions) and *recall* (the proportion of true positive predictions compared to the overall number of positive instances) : $F_1 = \frac{2 \times precision \times recall}{precision + recall}$. The F_1 score provides a balanced measure, combining these two performance aspects. It is therefore more informative compared to other metrics such as accuracy, especially when the training and testing sets are imbalanced [79], as in our case.

A central part of our approach to the training, validation, and verification of learning models is simulating the interactivity of visual analytics for real-time data, i.e. for use cases in which training data does not exist prior to user interaction. We assume the user (re)labels the incoming stream of data and therefore iteratively trains a model, which consequently meets their real-time needs. To replicate this process, we computationally evaluate the model's performance (as if it is successively trained by user-labeled data) by iteratively training the model with 10 new samples from the training dataset. We average the F_1 score obtained from each of these iterations and use the resulting number to measure the model's performance. In addition, we introduce a new variable, `window size`, for our training iterations. Specifically, due to the considerably small amount of training data provided by the user, we found that an appropriately small number of epochs (one forward and one backward pass over the training data in the model) was necessary to reduce performance degradation from initial overfitting. However, we also found that increasing the number of epochs could lead to higher F_1 scores as more data was provided. Thus, we use a sliding window of 110 samples that includes the (successively provided) new training data (10 samples) as well as the most recently used training data (100 samples) to both account for small amounts of training samples and increase the number of total training epochs for a given sample.

We use the validation data to optimize the hyperparameters for each of the CNN, LSTM, and RNN models. Specifically, after each training iteration with 10 new samples, we evaluate the neural network's F_1 score on the validation set to view its simulated performance as if it was trained by gradual user labeling. After identifying

the optimal hyperparameters for each of the CNN, LSTM, and RNN models, we evaluate their performance on the testing set.

Table 4.1 demonstrates the results from our validation stage. Specifically, it lists the average F_1 score obtained during each training simulation along with the total CPU time required to complete the simulation (accumulated with each training and evaluation iteration). Although in many applications, F_1 score alone is sufficient to evaluate machine learning models, it is not for ours. To see why, note that the LSTM model yields an F_1 score of 0.75, the highest of any hyperparameter combination. However, the LSTM model (with the highest F_1 score) takes approximately 4,242 seconds to complete training, whereas the CNN model (with the highest F_1 score) only takes 504 seconds. Thus, the LSTM model takes roughly eight times longer to simulate than the CNN model, but does not improve its F_1 score by a significant amount (LSTM: 0.75 vs. CNN: 0.74). In the context of interactive learning, we wish to balance the training/CPU time and performance such that the model both

Table 4.1.
Average precision, recall, F_1 score, and CPU time for the top three performing hyperparameter combinations on each of the CNN, LSTM, and RNN models. Bold numbers correspond to the highest F_1 scores and lowest CPU times for each of the three model types. We report the recall, precision, and F_1 score to four decimal places (when necessary) to distinguish the average F_1 scores.

Model	Learning Rate	Batch Size	Epochs	Dropout	Recurrent Dropout	Filter Size	Kernel Size	Optimizer	Average Precision	Average Recall	Average F_1 score	CPU Time (sec)
CNN	0.0079	10	1	–	–	16	2	Adam	0.75	0.73	**0.74**	**503.82**
CNN	0.01	50	2	–	–	16	2	Adagrad	0.73	0.71	0.72	522.47
CNN	0.0063	10	3	–	–	16	2	Adam	0.73	0.71	0.72	553.43
LSTM	0.0002	10	10	0.4	0.2	–	–	Adam	0.7597	0.7475	**0.7534**	4241.97
LSTM	0.0002	20	8	0.2	0.6	–	–	Adam	0.7597	0.7468	0.7530	**4100.37**
LSTM	0.0006	100	12	0.6	0.6	–	–	Adam	0.7559	0.7431	0.7493	4209.37
RNN	0.0001	10	7	0.0	0.2	–	–	Adam	0.7037	0.6957	**0.6996**	3069.81
RNN	0.0001	20	5	0.0	0.0	–	–	Adam	0.7028	0.6921	0.6973	**2805.52**
RNN	0.0001	100	12	0.0	0.2	–	–	Adam	0.70	0.69	0.69	3160.35

performs well and retrains in a short amount of time for rapid improvement (DG4). Therefore, it is necessary to consider both the CPU time and average F_1 score. With these optimization standards in mind, we chose the hyperparameters that yielded the highest F_1 scores for each model since the other hyperparameter combinations generated lower F_1 scores and higher or comparable CPU times. The selected combinations correspond to rows 1, 4, and 7 in Table 4.1 with the average F_1 scores in bold.

The testing process is identical to the validation process: after the model is trained with 10 new samples, its performance is measured by computing the average F_1 score on the testing set (using the optimized hyperparameters from the validation stage). Our results are summarized in Table 4.2. We found that the LSTM model yielded the highest F_1 score of 0.75. The CNN and RNN models achieved a 0.73 and 0.70 F_1 score, respectively. Based on these results and the previously discussed optimization standards, we selected the optimized CNN model for our classifier. In particular, the CNN simulation not only yielded a competitive average F_1 score of 0.73, but also achieved this score 6 to 8 times more quickly than the LSTM or RNN (Fig. 4.3), which is significant in terms of responding to user feedback in a timely manner.

The optimized CNN model yielded 0.74 and 0.73 average precision and recall scores respectively (Table 4.2, row 1). This model performance may be due to the initial lack

Table 4.2.
Testing results with the optimal hyperparameter combinations for the CNN, LSTM, and RNN models. The bold numbers correspond to the highest F_1 score and lowest CPU time among the three models.

Model	Average Precision	Average Recall	Average F_1 score	CPU Time (sec)
CNN	0.74	0.73	0.73	**501.10**
LSTM	0.76	0.74	**0.75**	4211.01
RNN	0.70	0.69	0.70	3085.59

of sufficient training data and difficulty in classifying certain tweets. For instance, after examining the testing dataset, we found that many misclassified tweets were extremely short (e.g., the tweet "screams internally" was misclassified as "Relevant") or contained complex disaster-related diction (e.g., the tweet "emergency dispatchers in boone county in the hot seat" was misclassified as "Relevant"). However, as we demonstrate in the next section, our model still outperforms state-of-the-art learning models on tweet datasets.

It is worth noting that we do not save the trained model from the validation or testing stages for evaluation in the next stage (or for use with SMART 2.0). We only save the optimized hyperparameters. This is because we assume that users start

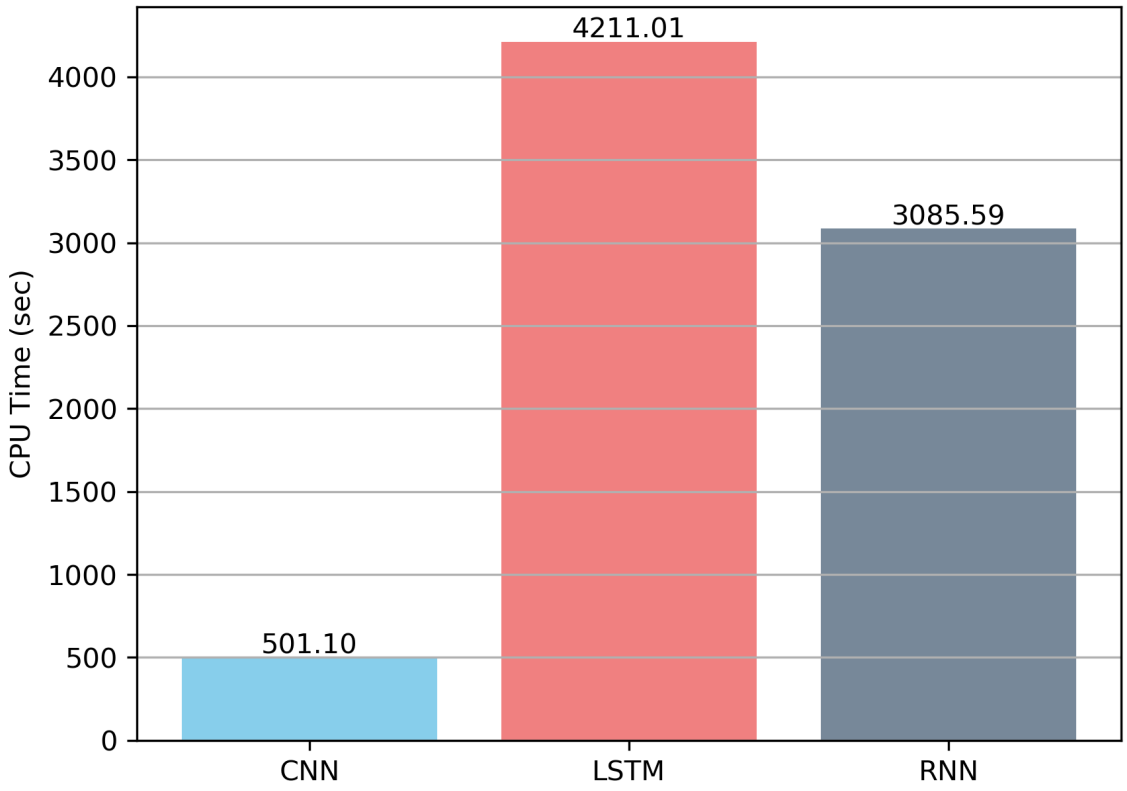

Figure 4.3. The total CPU time required for each model to complete the testing simulation. The CNN model is noticeably faster than both the LSTM and RNN models.

training a new model (for any event or topic they choose) by labeling the incoming stream of tweets.

In this section, we optimized the model on a sufficiently large dataset that contained tweets related to several types of disasters. In the next section, we evaluate the model on datasets containing tweets on specific events, which is representative of cases for situational awareness.

Evaluation

To further demonstrate the optimized CNN model's performance, we computationally evaluated it on wildfire, bombing, and train crash datasets from CrisisLexT26 [80], each of which contain approximately 1,000 tweets collected during 2012 and 2013 from 26 major crisis situations labeled by relevance. We apply a similar process to evaluate our optimized CNN model on these datasets as we did with the Figure Eight [77] dataset. Specifically, we split the data into 50% training and 50% testing sets (to replicate the experimental setting of To et al. [14], against which we will compare our results), train the model by supplying 10 tweets from the training set at a time (to simulate user labeling of streaming data), evaluate the resulting model on the entire testing set, and then average the F_1 scores for each evaluation.

We summarize our results in Table 4.3 and graph the model's performance for retraining with 10 new incoming tweets in Fig. 4.4, 4.5, and 4.6. In addition, we report the average CPU times to train the model during a single iteration (10 tweets) with each dataset in Table 4.3. Since the datasets vary slightly in size, we only compute the averages from the first 45 iterations since the smallest dataset (Boston Bombings) required 45 iterations to complete the simulation. We found that per-iteration training was fast and approximately 0.5 seconds with each dataset, which meets our timing demands (DG4).

We obtained 0.71, 0.64, and 0.88 F_1 scores for the Colorado wildfires, Boston bombings, and NY train crash datasets, respectively. Interestingly, the variance of

Table 4.3.
Average precision, recall, and F_1 score for three CrisisLexT26 [80] datasets.

Category	Average Precision	Average Recall	Average F_1 score	Average CPU Time (sec)
Colorado wildfires	0.72	0.71	0.71	0.49
Boston bombings	0.64	0.65	0.64	0.50
NY train crash	0.86	0.90	0.88	0.49

the F_1 scores over the datasets is significant. The textual data in the Boston bombings dataset, which yielded the lowest average F_1 score, was not as easy to separate into the different relevance categories by the model compared with the other two datasets. However, the F_1 score does eventually converge towards a higher value similar to the other datasets, indicating the potential presence of outliers during the first few training iterations. In addition, we found that the simulations converged to the average F_1 scores after training with approximately 190–230 tweets, depending on the dataset, meaning that users need to label 190–230 tweets to achieve the reported F_1 scores. However, the CrisisLexT26 datasets also correspond to specific events, such as wildfires. As such, we surmise that interactively training the model on specific, well-defined events will reduce the amount of training data needed to achieve satisfactory results than with generic constraints on relevance (e.g., a classifier about safety in general).

Finally, we compare our results with the learning-based algorithm employed by To et al. [14], who also evaluated their model's performance with CrisisLexT26 datasets. In particular, their learning-based approach used `Word2Vec`, TF-IDF, latent semantic indexing, and logistic regression for classifying data as relevant or irrelevant. The authors of that study split the dataset into two equal parts: one for training and one for testing. They trained the model once (as opposed to our iterative approach) and

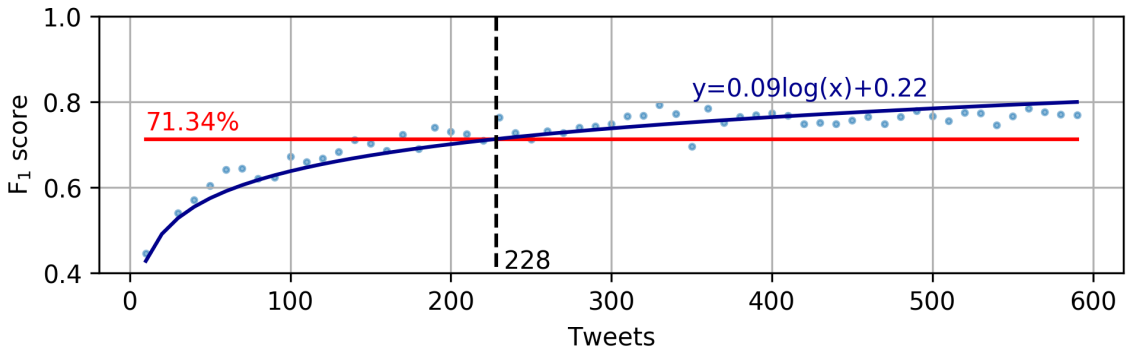

Figure 4.4. Optimized CNN F_1 score per training iteration of 10 tweets with the Colorado wildfires dataset (Table 4.3). The F_1 scores are logarithmically fitted and intersect with the average F_1 score (0.7134) at 228 tweets.

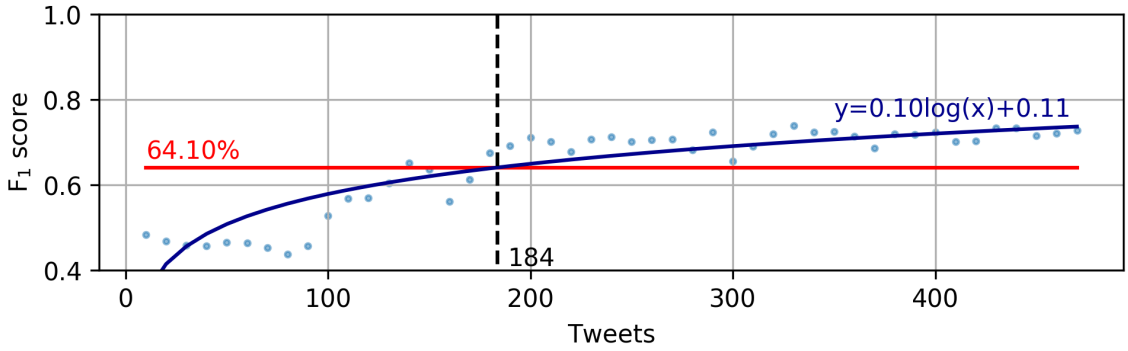

Figure 4.5. Optimized CNN F_1 score per training iteration of 10 tweets with the Boston bombings dataset (Table 4.3). The F_1 scores are logarithmically fitted and intersect with the average F_1 score (0.6410) at 184 tweets.

evaluated on the testing set. Their algorithm was able to yield high precision scores between 0.85–0.95, compared to our scores of 0.64–0.86. However, their recall scores were approximately 0.22–0.45, considerably lower than our recall scores of 0.65–0.90. Therefore, our approach outperforms the learning-based model presented by [14], in

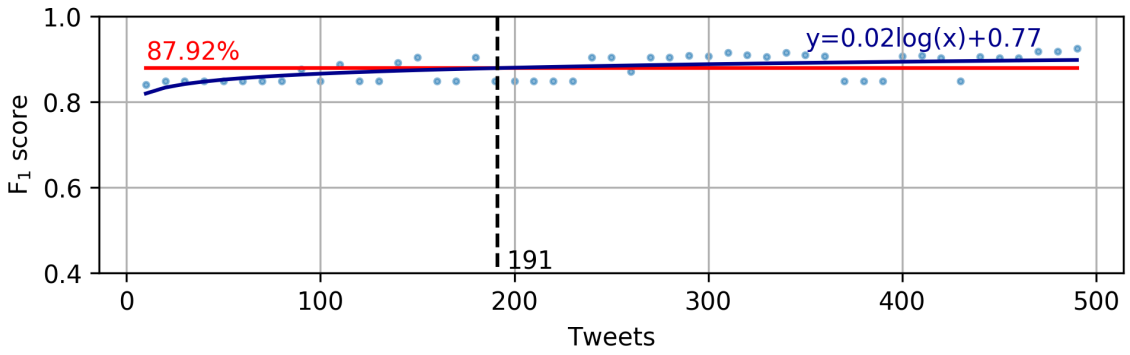

Figure 4.6. Optimized CNN F_1 score per training iteration of 10 tweets with the NY train crash dataset (Table 4.3). The F_1 scores are logarithmically fitted and intersect with the average F_1 score (0.8792) at 191 tweets.

terms of the overall F_1 score: our interactive approach achieves F_1 scores of 0.64–0.88 (depending on the dataset) compared to 0.45–0.64 by [14]. The authors also presented a matching-based approach that achieved a much higher F_1 score of 0.54–0.92, which is comparable to ours. However, they generate the set of hashtags to be used for matching by scanning all of the tweets in the dataset. Since we assume the data is streamed in real-time, and therefore, not available altogether, we use an iterative learning approach.

4.2 SMART 2.0

4.2.1 SMART

The Social Media Analytics and Reporting Toolkit (SMART) [11, 27] is a visual analytics application designed to support real-time situational awareness for first responders, journalists, government officials, and special interest groups. SMART obtains real-time publicly available geo-tagged data from the Twitter streaming API. The user is able to explore the trending and abnormal topics on various integrated

visualizations, including spatial topic model visualization and temporal views. The tweet time chart and theme river visuals convey the temporal distributions of topics if the user wishes to determine how the content of streamed social data has changed over time.

SMART uses string matching-based classifiers to visualize relevant data. Specifically, the user can either (a) select pre-defined filters, such as *Safety* or *Weather* (Fig. 4.7(c)), each using a series of related keywords for inclusion and exclusion of tweets in the subsequent topic-modeling (Fig. 4.7(f)) and (geo)visualizations (Fig. 4.7(b)), or (b) create their own filters by supplying keywords, and intersect or union multiple filters according to their needs. However, keyword-based matching is insufficient for finding relevant information as it fails to accurately capture semantic relevance and therefore effectively filter out noisy data. As an example, if the user were to apply the *Safety* classifier, it would be possible for the tweets "My house is on fire!" and "I was just fired from my job." to pass through the filter since they both include the keyword *fire*. However, the latter is unrelated to the intended semantic context of *Safety* and thus dilutes the filter's quality.

To address this problem, we integrate our interactive learning framework (the focus of this chapter) in the existing SMART application [11, 27] and seek domain expert feedback on the use of these models. We call the resulting extended application SMART 2.0. SMART 2.0 allows users to define string matching-based keyword filters (similar to SMART), but adds the ability for users to then iteratively refine and train the newly integrated models by labeling the filtered data as semantically relevant or not. In addition, the SMART 2.0 interface includes interactive visuals to facilitate user exploration, filtering, and refinement of relevant data (Fig. 4.7).

As with the model simulations in Section 4.1.3, SMART 2.0's underlying models are trained with successive batches of 10 user-labeled tweets. In cases where model predictions conflict with user labels, user labels override the model's since they represent the ground truth. In addition, users should not need to manually relabel the same data multiple times. Although conflicts might indicate that the model is not

sufficiently trained, the model trains with the same data during several successive iterations (as discussed in Section 4.1.3, par. 10), so conflicts might be resolved after future iterations.

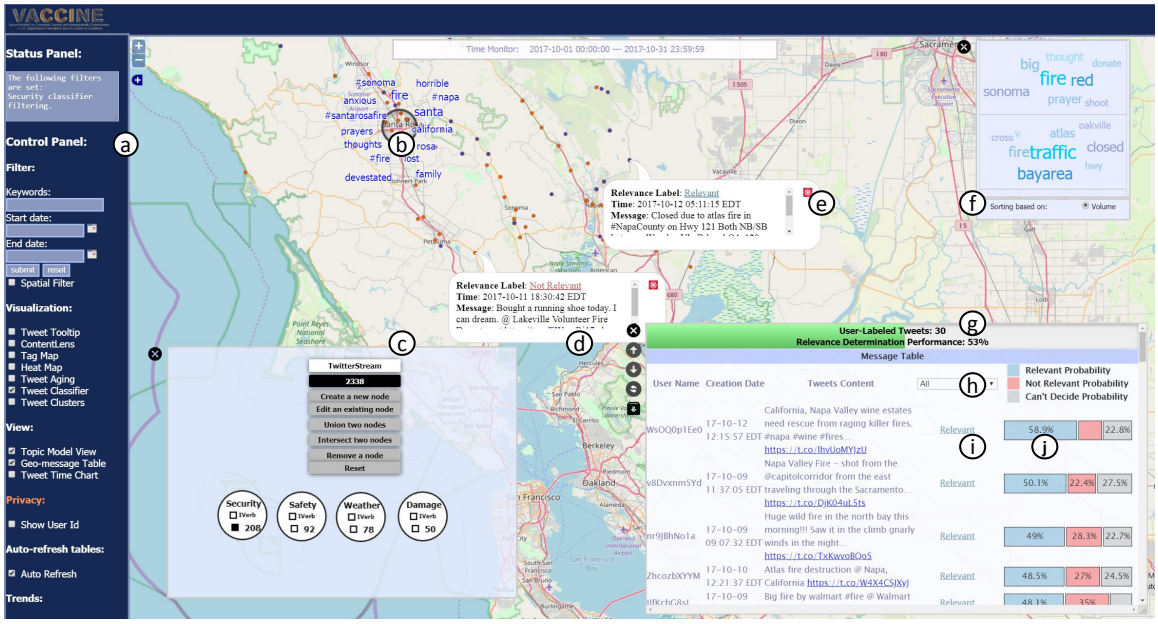

Figure 4.7. SMART 2.0 overview: (a) The control panel provides several filters, visualizations, and views. (b) The content lens visualization provides the most frequently used words within a selected area. (c) The tweet classifier visualization provides keyword-based filters to help reduce noisy data. (d)(e) Clicking a tweet on the map with the tweet tooltip visualization displays the tweet's time, message, and relevance label. (f) The topic-modeling view, based on Latent Dirichlet Allocation, extracts trending topics and the most frequently used words associated with each topic among tweets with specified relevancy. (g)–(j) The message table aggregates the tweets for efficient exploration with (g) the model's estimated classification performance (F_1 score), (h) a drop down box to filter data by their relevance labels, (i) color-coded relevance labels that can be changed by clicking on the label itself, and (j) associated relevance probabilities. Tweet map symbols are colored orange and purple to distinguish Twitter data from Instagram-linked tweets, respectively, since the latter contains potentially useful images for situational awareness.

4.2.2 SMART 2.0 Interface

The extensions to SMART 2.0's user interface, compared with SMART, concern the new interactive visuals that allow users to iteratively train machine learning models, utilize model predictions for rapid relevancy identification, and understand a model's reliability. The SMART 2.0 interface (Fig. 4.7) extends the interactive features of SMART for relevance identification in three primary ways:

1. Extending the tweet table (containing a tweet's creation date and text) by including the predicted relevance label, relevance label probabilities, label modification, model training performance, and relevance filtering.
2. Extending the interactive map containing the geo-tagged tweets whose relevancy can be individually inspected or modified.
3. Altering the content of existing SMART views (e.g., topic models and spatial topic lenses) using either all data or only relevant data (as identified by the model and corrected by the user).

In the following subsections, we discuss these three extensions in detail.

SMART 2.0: Table

The SMART 2.0 table (Fig. 4.7(g)–(j)) is extended from SMART in that it not only provides a tweet's creation date and text, but also provides the predicted relevance label (Fig. 4.7(i)) and the probabilities of a tweet belonging to any of the relevance classes (Fig. 4.7(j)) (DG1).

In particular, the relevance of a tweet can be "Relevant", "Not Relevant", or "Can't Decide". The "Relevant" label is colored blue, the "Not Relevant" label red, and the "Can't Decide" label gray to visually separate tweets with different relevance. SMART's preexisting blue color scheme motivated us to use the blue, red, and gray diverging coloring for relevancy in order to maintain visual appeal and harmony.

Users can directly click on relevance labels to correct the classifier's prediction (DG2). For instance, if a tweet is incorrectly marked "Relevant", clicking the label will change it to "Not Relevant" or "Can't Decide", depending on the label the user wishes to assign. Further, a drop down box is included at the top of the relevance label column (Fig. 4.7(h)), which provides the option to filter out data that does not have a specified relevancy (DG1). For example, by selecting "Relevant" from the drop down box, the table will remove tweets with labels "Not Relevant" and "Can't Decide" from all views and visualizations in SMART, including geovisualizations and temporal views.

The table also displays the *degree* (or confidence) of a tweet's relevancy. In specific, the probabilities of a tweet being "Relevant", "Not Relevant", or "Can't Decide" are represented as a horizontal segmented bar graph and sized proportional to their respective percentages (Fig. 4.7(j)). In addition, the user can sort tweets based on relevancy probability in ascending or descending order.

We provide the relevance probabilities and associated sorting actions as a supplementary relevance filtering mechanism (DG1). In particular, it is possible for tweets to be classified as "Relevant" by the model, for example, but with low confidence. The probability filtering allows the user to specifically view high-confidence relevant data and therefore further reduce potentially noisy data.

The table provides a performance bar that encodes the estimated performance (F_1 score) of the underlying learning model (Fig. 4.7(g)), as well as the number of user-labeled tweets, to inform the user of the model reliability. Since labeled testing data is not available to evaluate the model for real-time training (because we assume the user may train on any type of event data and has their own specifications for relevancy), the model's performance can only be estimated. Based on our evaluations in Section 4.1.3, par. 17, with datasets typical of situational awareness scenarios (Table 4.3), the Colorado wildfires dataset generated the F_1 score (0.71) closest to the average of the three datasets (0.74). Therefore, we use the Colorado wildfires

dataset's logarithmic trendline $y = 0.09 \log_e(x) + 0.22$ (Fig. 4.4) to approximate the model's F_1 score as a function of the number of user-labeled tweets.

SMART 2.0: Map

The SMART 2.0 map is extended from SMART in that it includes a tweet's relevance label (which can be modified) in addition to its text and creation date (Fig. 4.7(d)(e)). Through the *Tweet Tooltip*, the user can directly click on tweet symbols on the map to view their text and associated relevancy (DG1). In addition, the user can correct the classified relevance label (DG2) by clicking on the label itself. Map inspection can allow the user to view and investigate potential geographical relevancy trends. For example, during crisis events, relevant tweets might be closely grouped on the map, so it may be more beneficial for the user to view predicted relevance from the map itself.

The interactions between the table and map are synchronized. If the user relabels data on the map, the associated new label will also be updated in the table, and vice versa. In addition, selecting a relevancy filter from the drop down box in the table filters the tweets on the map.

SMART 2.0: Integration with Existing Visualizations

Many of SMART's original visualizations, such as the topic-model views, spatial topic lenses, and temporal views help users make sense of spatiotemporal text data. Therefore, we integrated all of these views in SMART 2.0 with the relevance extensions.

Users have the option to view only relevant or all the data (including irrelevant tweets) in various visualizations in case the interactive classifiers are not yet trained to desirable accuracies since, as we show in Section 4.1.3, par. 19, classifiers typically require around 200 user-labeled tweets to achieve F_1 scores of 0.70–0.80. If they choose to view only relevant tweets, any relevance filtering action also updates the data used

by other visuals. For example, the topic-modeling view (Fig. 4.7(f)) extracts the top 10 topics from the tweets and displays the most frequently used words for each topic. If the user filters out irrelevant tweets, the topic-modeling view will only be applied to the remaining relevant tweets. It is important to note that the majority of visualizations in SMART 2.0 require a minimum number of tweets in order to render. When filtered relevant data is scarce, visualizations do not populate, in which case users can individually inspect tweets. For instance, the topic-modeling view requires at least 10 tweets to extract topics.

Overall, SMART 2.0's suite of visualization tools can be used in combination with relevance interactions to further understand trends and important spatiotemporal characteristics of relevant data.

4.3 User Experience

In this section, we provide usage scenarios and feedback from domain experts that demonstrate our framework's effectiveness and usability.

4.3.1 Usage Scenario 1

Alice is an emergency dispatcher interested in identifying people in need for help or hazardous locations during a hurricane. She uses SMART 2.0 to find any related social media posts near the affected area. She adds a new filter *Hurricane* and provides an appropriate set of filter keywords such as "hurricane", "help", "blocked", and "trapped".

After applying the *Hurricane* filter, she explores the filtered tweets in the table and finds a tweet labeled "Relevant" that says "Does anyone know how to get help setting up my TV?". Since the tweet is unrelated to a hurricane, she relabels it as "Not Relevant". After further browsing the table, she finds a tweet that says "The road near Taylor Loop is blocked from a broken tree.", but it is labeled as "Not Relevant". Since the tweet contains actionable information, she relabels it as

"Relevant". After labeling several more tweets for model training and noticing that the model predicts correctly, she decides to only view "Relevant" tweets and sort them by most relevant. She promptly identifies a tweet posted only a few minutes ago marked as highly relevant. It reads "Car just crashed into tree blocking road near Taylor Loop!". Alice immediately notifies first responders of the location to provide assistance.

By using SMART 2.0, Alice is able to identify important, relevant data more quickly through interactively training the model to remove noise and then filtering by relevance.

4.3.2 Usage Scenario 2

To demonstrate the generalizability of our framework to other domains, we applied our interactive framework in real-time during the Purdue vs. Virginia 2019 March Madness game in the Kentucky area. We assumed the role of a journalist who wanted to follow public discourse on the game by identifying the relevant tweets. We first constructed a *Sports* filter, which included keywords such as "Purdue", "game", "score", and "#MarchMadness". We then interacted with the streaming data by iteratively labeling the relevancy of tweets (from scratch) and found that the system correctly classified incoming data after roughly 80 training samples (Fig. 4.8). We noticed that the time intervals between successive trainings increased, indicating that it was more difficult to find incorrectly labeled data towards the end and that the model gradually learned from user feedback. In particular, the interval between the first and second training iterations was 2 minutes, whereas the interval between the final two was 4 minutes.

4.3.3 Domain Expert Feedback

We piloted SMART 2.0 with two groups of first responders, each containing two individuals, who frequently use SMART during events for situational awareness in their

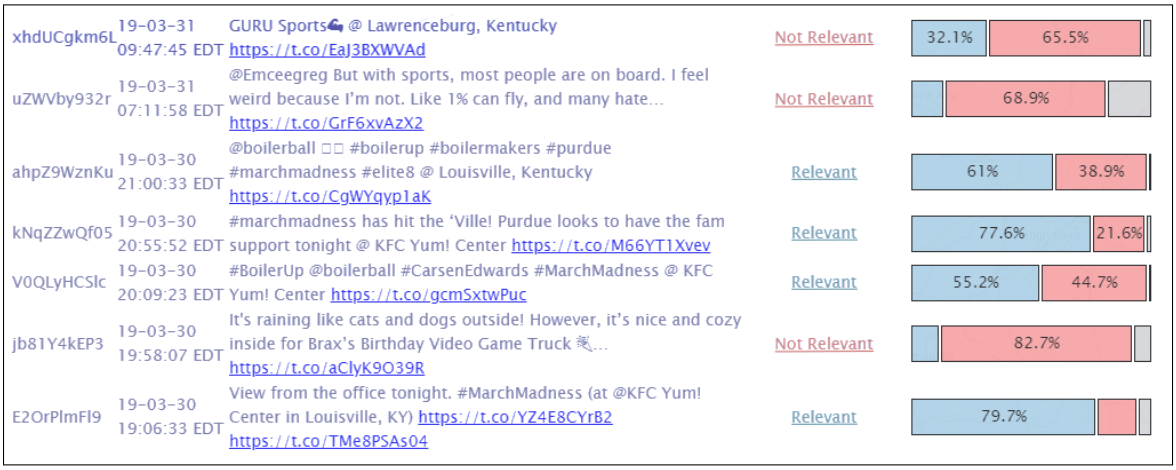

Figure 4.8. Tweets with correctly predicted relevancy from the Purdue vs. Virginia 2019 March Madness game after the user (re)labels 80 tweets.

operations. Both groups participated in separate 1-hour long sessions via conference call in which they iteratively trained a classifier from scratch and applied relevance filtering and visualizations to assess the implemented framework. They received a tutorial of SMART 2.0 30 minutes beforehand and were provided with web access to the system to complete the session. For both groups, we simulated the real-time use of SMART 2.0 by feeding in a stream of historical data on events (previously collected). For the first group, the system presented unlabeled tweets from the Las Vegas shooting on October 1, 2017, in the Las Vegas area. For the second group, we used unlabeled tweets from the October 2017 Northern California wildfires. We used historical event datasets to ensure the existence of sufficient training relevant samples for a situational awareness scenario.

The domain experts in the first group applied the *Safety*, *Damage*, and *Security* filters during the iterative training process, resulting in 317 tweets. They trained on the same underlying model for all three filters, as they considered them semantically related. In total, after relabeling approximately 200 tweets, they indicated that they could trust the model to predict accurately and were pleased that the tweets they had

not seen before from the *Security* filter were labeled correctly. Their definition for relevancy was tweets containing actionable information. For instance, they marked tweets containing information about road closures, blood drive locations, or death counts as relevant. They labeled data with general comments regarding the shooting, such as "I hope everyone is safe now...terrible shooting...", as irrelevant since they did not provide actionable information. Interestingly, they also marked tweets that would influence public opinion (and therefore may cause action) such as those from bots or trolls as relevant since they still contained actionable information.

The domain experts from the second group followed a similar process in which they applied the *Safety*, *Damage*, and *Security* filters, resulting in 445 tweets, and trained a learning model for relevance. They found that after training on roughly 67 tweets, the model satisfactorily predicted relevancy. As with the first group, these domain experts labeled tweets as relevant if they contained actionable information.

The domain experts from both groups found SMART 2.0 to be easy to use and effective in identifying important data. For instance, they discovered relevant, actionable information after training the model: specific blood drive locations to aid shooting victims. Notably, the users mentioned that they found themselves relabeling less data as training progressed since the system provided more correct labels. They were pleased that they had the option to view only relevant data, but could see all of the data regardless of relevancy to avoid potentially missing important misclassified data, and that the model was responsive to user training. In addition, they found the relevance percentage bars to be helpful in determining the tweets that were potentially the most relevant.

One concern the domain experts had was that SMART 2.0 does not indicate the number of tweets that are predicted as relevant. They felt this extension could help them infer the occurrence of events or potential crises. For example, the number of relevant tweets for the *Safety* classifier would likely increase significantly during a widespread disaster. We plan to introduce this feature in the next development cycle. However, we have added a visualization of estimated model performance in

SMART 2.0 (Fig. 4.7(g)) to help users ascertain the reliability of a model's relevancy predictions.

Overall, the feedback from the domain experts was positive and helpful, indicating the system's practicality and usefulness in facilitating real-time situational awareness. In addition, they have asked to use SMART 2.0 in their emergency operations center.

4.4 Discussion

Our interactive learning framework and SMART 2.0 integration were developed with the user in mind, influencing all of our design, computational evaluation, and implementation choices. Our user-centered model and SMART 2.0 application contribute to both the machine learning and visual analytics communities. We bridge the two fields by demonstrating how models can be interactively trained and evaluated while keeping the user in mind, and used to facilitate situational awareness for real-life, practical use.

The scalability of our framework is a natural concern, especially since SMART and many deployed real-time visual analytics applications contain multiple users who require responsive interfaces while monitoring crisis events. We deliberately designed the framework architecture with scalability in mind. As mentioned in Section 4.1.3, pars. 12–13, we selected the model and optimal hyperparameters based on training/CPU time in an effort to maximize the model's computational speed. Further, SMART 2.0 filters and views *at most* 800–900 tweets at a time, although user-specified filtering (typical in situational awareness scenarios) reduces the data to only a few hundred tweets, as demonstrated in Section 4.3.3. It takes only 2–3 seconds to calculate and retrieve their relevance labels over the network from the server where the model resides, and per-iteration training is fast, as established in Section 4.1.3, par 18.

Finally, although we rigorously optimize and evaluate our machine learning model, the hyperparameter combinations were only tuned with the Figure Eight dataset [77].

Since optimal hyperparameters can depend on the dataset itself, it is possible our model may not be optimally tuned for different datasets, even though that optimization may be negligible from a user standpoint. We did use other datasets in our model evaluation to show the satisfactory resulting performance (Section 4.1.3, par. 17). Given that the Figure Eight dataset classifies generic events as relevant or irrelevant, as opposed to specific events, we expect that our model performs well on many different event types.

4.5 Summary

In this chapter, we presented a novel interactive framework in which users iteratively (re)train neural network models with streaming text data in real-time to improve the process of finding relevant information. We optimized and evaluated a machine learning model with various datasets related to situational awareness and adapted the model to learn at interactive rates. According to evaluation results, our model outperforms state-of-the-art learning models used in similar classification tasks. Finally, we integrated our framework with the SMART application and extended it to SMART 2.0, allowing users to interactively explore, identify, and refine tweet relevancy to support real-time situational awareness. Our discussions with multiple first responders who use SMART 2.0 indicated positive feedback and user experience. In particular, their assessments demonstrated that our interactive framework significantly improved the time-consuming process of finding crucial information during real-time events.

5 CONCLUSION AND FUTURE WORK

In summary, this work focused on two barriers for social media analysts and first responders: lack of explicitly geotagged data and difficulty in identifying relevant information from large, noisy data streams. We provided techniques grounded in visual analytics as solutions for both problems. To reiterate, the contributions of this work can be summarized as follows:

1. **A geolocation inference approach for real-time visual analytics [68].** We adapted, improved, and evaluated an existing deep learning geolocation prediction model. We further integrated the improved model with an interactive visual analytics system tailored for real-time situational awareness. We experimented with social media data streams, demonstrating noticeable increases in the amount of geotagged data, therefore providing improved situational awareness.
2. **An interactive learning approach for identifying relevant streaming data [10].** We provided a novel interactive learning framework that allows users to train relevance classifiers with real-time streaming data for improved situational awareness. We evaluated the performance and computational efficiency of our framework with different neural networks. We further evaluated the selected model on several crisis-related datasets and showed that it outperforms state-of-the-art machine learning classifiers. We provided SMART 2.0, a visual analytics application that realizes our interactive learning framework, and demonstrated its effectiveness with usage scenarios and domain expert feedback.

Our future work can be summarized from the following perspectives:

- **Geolocation inference:** We plan to explore additional enhancements to deepgeo2 from recent work, such as Bayesian models [8], and investigate alternative

embedding methods, such as combining character and word embeddings. Further, we will investigate human-in-the-loop methods to improve the geolocation prediction. For instance, it might be possible for users to explicitly provide city labels of non-geotagged tweets, or correct predicted city labels. We will also enhance the cartographic representation of SMART for visualizing precise and estimated tweet locations.

- **Supporting multilingual data with our interactive learning framework:** SMART 2.0 currently only collects English tweets, although supporting non-English languages (one at a time) with our current design (e.g., Spanish only) is straightforward since `Word2Vec` embeddings can be independently trained on a corpus in the target language [81]. Extending our system to support multilingual tweets would be a powerful asset, especially for multilingual users, in amplifying situational awareness by leveraging relevancy of tweets issued in different languages. However, the multilingual model performance evaluation and testing is an open area for research. In addition, determining the specifics of how a single relevance classifier might be trained with multilingual tweets requires careful attention. For instance, training iterations with Spanish tweets should also affect the relevancy of semantically-related English tweets. Since similar words in different languages likely have different vector representations (embeddings), multilingual mappings must be learned or training must be performed differently, such as with parallel corpora [81]. Multilingual support also requires changes in SMART 2.0's language-dependent visualizations, such as the topic-modeling view (Fig. 4.7(f)). Translation to a unified language or extracting topics separately for each language are two potential solutions.
- **Accommodating situations with scarce relevant data:** Training a model during a particular event, such as a disaster, can be straightforward due to potentially larger amounts of relevant data. However, social media data during periods without major events are likely to contain very few, if any, relevant tweets. As such, if the user *only* trains the model on irrelevant data, it will

poorly predict relevant data since it has only be taught what is irrelevant. Although the user can improve the model through training during a real-life disaster, they are required to know when and where the disaster occurs to begin training. This can be problematic if the user wishes to rely on relevancy predictions to detect hazardous situations.

To accommodate time periods in which relevant data is scarce, we plan to introduce an interactive feature in which users can provide example tweets or external resources for specific relevance labels. For instance, if the user would like to train a *Hurricane* classifier before a hurricane event, they could provide a relevant text such as "I'm stranded by this hurricane. Please help!". The model could then detect relevant tweets once the hurricane begins as opposed to requiring user training during the event. In addition, we plan to provide the user with the option to visit specific historical data to train existing classifiers, as done by Bosch et al. [24]. Furthermore, although SMART 2.0 provides integrated visuals and filtering and labeling interactions to quickly identify relevant tweets, supplementary high-level visualizations of relevant data can increase understanding of overall trends. We plan to introduce additional map layers, such as a heat map, to better understand geographical trends of relevant data. For instance, visually clustered groups of relevant data during a disaster event can alert the user to specific affected areas that might need attention or contain actionable information.

- **Investigating the interactive learning performance on generic classifiers:** Our interactive learning performed well on target datasets (i.e., wildfire, bombing, and crash) as explained in Section 4.1.3, par 17. Specifically, it required the users to label approximately 200 tweets to achieve acceptable F_1 scores. However, tweets are short, and therefore, more research is required to investigate the suitability of our approach for "general" classifiers, such as ones that learn to classify relevant data to "safety". As safety can be affected

by many events or situations, the model may need additional training that is typical of a targeted dataset.

- **New interactive learning techniques:** We will research alternative interactive methods for training relevance classifiers. Our current framework relies on users to individually inspect and (re)label data, which can be burdensome. One potential alternative is allowing users to label data subsets through learned high-level concept representations.

REFERENCES

REFERENCES

[1] María Martínez-Rojas, María Del Carmen Pardo-Ferreira, and Juan Carlos Rubio-Romero. Twitter as a tool for the management and analysis of emergency situations: A systematic literature review. *International Journal of Information Management*, 43:196–208, 2018.

[2] Luke S. Snyder, Morteza Karimzadeh, Christina Stober, and David S. Ebert. Situational awareness enhanced through social media analytics: A survey of first responders. In *2019 IEEE International Symposium on Technologies for Homeland Security (HST)*, 2019.

[3] Jens A de Bruijn, Hans de Moel, Brenden Jongman, Jurjen Wagemaker, and Jeroen CJH Aerts. Taggs: Grouping tweets to improve global geoparsing for disaster response. *Journal of Geovisualization and Spatial Analysis*, 2(1):2, 2018.

[4] Kisung Lee, Raghu Ganti, Mudhakar Srivatsa, and Prasant Mohapatra. Spatio-temporal provenance: Identifying location information from unstructured text. In *2013 IEEE International Conference on Pervasive Computing and Communications Workshops*, 2013.

[5] Tien Huu Do, Duc Minh Nguyen, Evaggelia Tsiligianni, Bruno Cornelis, and Nikos Deligiannis. Multiview deep learning for predicting twitter users' location. *arXiv preprint arXiv:1712.08091*, 2017.

[6] Abhinav Kumar and Jyoti Prakash Singh. Location reference identification from tweets during emergencies: A deep learning approach. *International Journal of Disaster Risk Reduction*, 33:365–375, 2019.

[7] Jey Han Lau, Lianhua Chi, Khoi-Nguyen Tran, and Trevor Cohn. End-to-end network for twitter geolocation prediction and hashing. In *International Joint Conference on Natural Language Processing*, 2017.

[8] Pengfei Li, Hua Lu, Nattiya Kanhabua, Sha Zhao, and Gang Pan. Location inference for non-geotagged tweets in user timelines. *IEEE Transactions on Knowledge and Data Engineering*, 31(6):1150–1165, 2019.

[9] Stuart E. Middleton, Lee Middleton, and Stefano Modafferi. Real-time crisis mapping of natural disasters using social media. *IEEE Intelligent Systems*, 29(2):9–17, 2014.

[10] Luke S. Snyder, Yi-Shan Lin, Morteza Karimzadeh, Dan Goldwasser, and David S. Ebert. Interactive learning for identifying relevant tweets to support real-time situational awareness. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):558–568, 2020.

[11] Jiawei Zhang, Junghoon Chae, Chittayong Surakitbanharn, and David S Ebert. SMART: Social media analytics and reporting toolkit. In *The IEEE Workshop on Visualization in Practice 2017*, pages 1–5, 2007.

[12] Hila Becker, Mor Naaman, and Luis Gravano. Beyond trending topics: Real-world event identification on twitter. In *Proceedings of the 5th International Conference on Weblogs and Social Media*, pages 438–441, 2011.

[13] Z. Ashktorab, C. Brown, M. Nandi, and A. Culotta. Tweedr: Mining twitter to inform disaster response. In *Proceedings of the 11th International ISCRAM Conference*, pages 354–358, 2014.

[14] Hien To, Sumeet Agrawal, and Cyrus Shahabi. On identifying disaster-related tweets: Matching-based or learning-based? In *IEEE 3rd International Conference on Multimedia Big Data (BigMM)*, 2017.

[15] N Diakopoulos, M Naaman, and F Kivran-Swaine. Diamonds in the rough: Social media visual analytics for journalistic inquiry. In *IEEE Symposium on Visual Analytics Science and Technology*, pages 115–122, 2010.

[16] Samujjwal Ghosh and Maunendra Sankar Desarkar. Class specific TF-IDF boosting for short-text classification: Application to short-texts generated during disasters. In *Companion of the The Web Conference 2018 (WWW '18)*, pages 1629–1637, 2018.

[17] Sarvnaz Karimi, Jie Yin, and Cecile Paris. Classifying microblogs for disasters. In *Proceedings of the 18th Australasian Document Computing Symposium*, ADCS '13, pages 26–33, 2013.

[18] Tahora H Nazer, Fred Morstatter, Harsh Dani, and Huan Liu. Finding requests in social media for disaster relief. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1410–1413, 2016.

[19] Dat Tien Nguyen, Shafiq Joty, Muhammad Imran, Hassan Sajjad, and Prasenjit Mitra. Applications of online deep learning for crisis response using social media information. *arXiv preprint arXiv:1610.01030*, 2016.

[20] Dat Tien Nguyen, Kamela Ali Al Mannai, Shafiq Joty, Hassan Sajjad, Muhammad Imran, and Prasenjit Mitra. Rapid classification of crisis-related data on social networks using convolutional neural networks. *arXiv preprint arXiv:1608.03902*, 2016.

[21] Koustav Rudra, Subham Ghosh, Niloy Ganguly, Pawan Goyal, and Saptarshi Ghosh. Extracting situational information from microblogs during disaster events: a classification-summarization approach. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015.

[22] Fujio Toriumi and Seigo Baba. Real-time tweet classification in disaster situation. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 117–118, 2016.

[23] Tommaso Zoppi, Andrea Ceccarelli, Francesco Lo Piccolo, Paolo Lollini, Gabriele Giunta, Vito Morreale, and Andrea Bondavalli. Labelling relevant events to support the crisis management operator. *Journal of Software: Evolution and Process*, 30(3), 2018.

[24] Harald Bosch, Dennis Thom, Florian Heimerl, Edwin Puttmann, Steffen Koch, Robert Kruger, Michael Worner, and Thomas Ertl. Scatterblogs2: Real-time monitoring of microblog messages through user-guided filtering. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2022–2031, 2013.

[25] F Heimerl, S Koch, H Bosch, and T Ertl. Visual classifier training for text document retrieval. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2839–2848, 2012.

[26] Burr Settles. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, pages 1467–1478, 2011.

[27] Jiawei Zhang, Shehzad Afzal, Dallas Breunig, Jing Xia, Jieqiong Zhao, Isaac Sheeley, Joseph Christopher, David S Ebert, Chen Guo, Shang Xu, Jim Yu, Qiaoying Wang, Chen Wang, Zhenyu Qian, and Yingjie Chen. Real-time identification and monitoring of abnormal events based on microblog and emergency call data using SMART. In *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 393–394, 2014.

[28] Catherine D'Ignazio, Rahul Bhargava, Ethan Zuckerman, and Luisa Beck. Cliffclavin: Determining geographic focus for news. *NewsKDD: Data Science for News Publishing*, 2014.

[29] Morteza Karimzadeh, Scott Pezanowski, Alan M. MacEachren, and Jan O. Wallgrün. GeoTxt: A scalable geoparsing system for unstructured text geolocation. *Transactions in GIS*, 23(1):118–136, 2019.

[30] Ozer Ozdikis, Halit Oğuztüzün, and Pinar Karagoz. A survey on location estimation techniques for events detected in twitter. *Knowledge and Information Systems*, 52(2):291–339, 2016.

[31] David Jurgens, Tyler Finethy, James McCorriston, Yi Tian Xu, and Derek Ruths. Geolocation prediction in twitter using social networks: A critical analysis and review of current practice. In *Proceedings of the 9th International Conference on Web and Social Media*, 2015.

[32] Yujie Qian, Jie Tang, Zhilin Yang, Binxuan Huang, Wei Wei, and Kathleen M Carley. A probabilistic framework for location inference from social media. *arXiv preprint arXiv:1702.07281*, 2017.

[33] Dennis Thom, Harald Bosch, Robert Krueger, and Thomas Ertl. Using large scale aggregated knowledge for social media location discovery. In *2014 47th Hawaii International Conference on System Sciences*, 2014.

[34] Nghia Duong-Trung, Nicolas Schilling, and Lars Schmidt-Thieme. Near real-time geolocation prediction in twitter streams via matrix factorization based regression. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, 2016.

[35] Sanae Mahtal, Cristina Lupu, Benedikt Armbruster, Marvin Bechtold, Maximilian Reichel, Thomas Wangler, Dennis Thom, Steffen Koch, and Thomas Ertl. Analytical workbench for integrated social media geo-inference. In *VGI Geovisual Analytics Workshop*, 2018.

[36] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Computational Linguistics*, 35(3):399–433, 2009.

[37] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12), 2009.

[38] Marco Pennacchiotti and Ana-Maria Popescu. A machine learning approach to twitter user classification. In *5th International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2011.

[39] X.-H. Phan, L.-M. Nguyen, and S. Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th International Conference on World Wide Web*, pages 91–99, 2008.

[40] Mengen Chen, Xiaoming Jin, and Dou Shen. Short text classification improved by learning multi-granularity topics. In *22nd International Joint Conference on Artificial Intelligence*, pages 1776–1781, 2011.

[41] Xia Hu, Nan Sun, Chao Zhang, and Tat-Seng Chua. Exploiting internal and external semantics for the clustering of short texts using world knowledge. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 919–928, 2009.

[42] Pak C. Wong, Elizabeth G. Hetzler, Christian Posse, Mark A. Whiting, Susan L. Havre, Nick O. Cramer, Anuj Shah, Mudita Singhal, Alan E. Turner, and Jim Thomas. IN-SPIRE InfoVis 2004 contest entry. In *Proceedings of the IEEE Symposium on Information Visualization*, 2004.

[43] Lap Trieu, Huy Tran, and Minh-Triet Tran. News classification from social media using twitter-based doc2vec model and automatic query expansion. In *Proceedings of the 8th International Symposium on information and communication technology*, pages 460–467, 2017.

[44] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[45] W. Ling, C. Dyer, A. Black, and I. Trancoso. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304, 2015.

[46] Joseph Lilleberg, Yun Zhu, and Yanqing Zhang. Support vector machines and word2vec for text classification with semantic features. In *IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCICC)*, pages 136–140, 2015.

[47] Omar Abdelwahab and Adel Elmaghraby. UofL at SemEval-2016 task 4: Multi domain word2vec for twitter sentiment classification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 164–170, 2016.

[48] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1555–1565, 2014.

[49] D. Tang, B. Qin, and T. Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1422–1432, 2015.

[50] Muhammad Imran, Prasenjit Mitra, and Carlos Castillo. Twitter as a lifeline: Human-annotated twitter corpora for NLP of crisis-related messages. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC)*, 2016.

[51] Adam Marcus, Michael Bernstein, Osama Badar, David Karger, Samuel Madden, and Robert Miller. TwitInfo: aggregating and visualizing microblogs for event exploration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 227–236, 2011.

[52] Scott Pezanowski, Alan M Maceachren, Alexander Savelyev, and Anthony C Robinson. SensePlace3: A geovisual framework to analyze place—time–attribute information in social media. *Cartography and Geographic Information Science*, 45(5):420–437, 2018.

[53] Jennifer G Dy and Carla E Brodley. Visualization and interactive feature selection for unsupervised data. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 360–364, 2000.

[54] Emily Wall, Subhajit Das, Ravish Chawla, Bharath Kalidindi, Eli T Brown, and Alex Endert. Podium: Ranking data using mixed-initiative visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):288–297, 2018.

[55] J. Bernard, M. Hutter, M. Zeppelzauer, D. Fellner, and M. Sedlmair. Comparing visual-interactive labeling with active learning: An experimental study. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):298–308, 2018.

[56] Justin Talbot, Bongshin Lee, Ashish Kapoor, and Desney Tan. EnsembleMatrix: interactive visualization to support machine learning with multiple classifiers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1283–1292, 2009.

[57] M. Kahng, P.Y. Andrews, A. Kalro, and D.H.P. Chau. ActiVis: Visual exploration of industry-scale deep neural network models. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):88–97, 2018.

[58] Dominik Sacha, Matthias Kraus, Daniel A Keim, and Min Chen. VIS4ML: An ontology for visual analytics assisted machine learning. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):385–395, 2019.

[59] D. Sacha, M. Sedlmair, L. Zhang, J. Aldo Lee, D. Weiskopf, S. North, and D. Keim. Human-centered machine learning through interactive visualization: Review and open challenges. In *Proceedings of the 24th European Symposium on Artificial Neural Networks (ESANN)*, pages 641–646, 2016.

[60] Alex Endert, Patrick Fiaux, and Chris North. Semantic interaction for visual text analytics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 473–482, 2012.

[61] Burr Settles. Active learning literature survey (computer sciences technical report 1648). *University of Wisconsin-Madison*, 2009.

[62] Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, 2000.

[63] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, 1994.

[64] Kostiantyn Kucher, Carita Paradis, Magnus Sahlgren, and Andreas Kerren. Active learning and visual analytics for stance classification with alva. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 7(3):1–31, 2017.

[65] Mayank Kejriwal and Yao Gu. A pipeline for rapid post-crisis twitter data acquisition, filtering and visualization. *Technologies*, 7(2):33, 2019.

[66] Gabriela Bosetti, Elöd Egyed-Zsigmond, and Lucas Okumura Ono. Cati: An active learning system for event detection on mibroblogs' large datasets. In *15th International Conference on Web Information Systems and Technologies (WEBIST)*, 2019.

[67] Xia Hu, Jiliang Tang, Huiji Gao, and Huan Liu. Actnet: Active learning for networked texts in microblogging. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 306–314, 2013.

[68] Luke S. Snyder, Morteza Karimzadeh, Ray Chen, and David S. Ebert. City-level geolocation of tweets for real-time visual analytics. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery (GeoAI)*, 2019.

[69] Word2vec. https://code.google.com/archive/p/word2vec.

[70] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[71] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[72] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[73] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[74] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of CNN and RNN for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.

[75] S. Lai, L. Xu, K. Liu, and J. Zhao. Recurrent convolutional neural networks for text classification. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, volume 3, pages 2267–2273, 2015.

[76] Keras. https://keras.io.

[77] Figure eight: Disasters on social media. https://www.figure-eight.com/data-for-everyone/.

[78] M. Sokolova, N. Japkowicz, and S. Szpakowicz. Beyond accuracy, F-score and ROC: A family of discriminant measures for performance evaluation. *AI 2006: Advances in Artificial Intelligence*, pages 1015–2021, 2006.

[79] Andrea Dal Pozzolo, Olivier Caelen, Reid A Johnson, and Gianluca Bontempi. Calibrating probability with undersampling for unbalanced classification. In *IEEE Symposium Series on Computational Intelligence*, pages 159–166, 2015.

[80] Alexandra Olteanu, Sarah Vieweg, and Carlos Castillo. What to expect when the unexpected happens: Social media communications across crises. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '15, pages 994–1009, 2015.

[81] Alexandre Bérard, Christophe Servan, Olivier Pietquin, and Laurent Besacier. MultiVec: a multilingual and multilevel representation learning toolkit for NLP. In *The 10th edition of the Language Resources and Evaluation Conference (LREC)*, 2016.

VITA

VITA

Luke S. Snyder is a MS student in the Department of Computer Science at Purdue University. His research interests broadly include visual analytics, visualization, interactive machine learning, and methods for situational awareness. Luke holds a BS in Computer Science from the University of Arkansas. Contact him at snyde238@purdue.edu.