2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Society Track

Crowdsourced Exploration of Mobile App Features

A Case Study of the Fort McMurray Wildfire

Maleknaz Nayebi, Rachel Quapp, Guenther Ruhe
SEDS laboratory
University of Calgary
Calgary, Canada
Email: {mnayebi, rmquapp, ruhe}@ucalgary.ca,

Mahshid Marbouti, Frank Maurer
ASE labratory
University of Calgary
Calgary, Canada
Email: {mmarbout, fmaurer}@ucalgary.ca

***Abstract*—The ubiquity of mobile devices has led to unprecedented growth in not only the usage of apps, but also their capacity to meet people's needs. Smart phones take on a heightened role in emergency situations, as they may suddenly be among their owner's only possessions and resources. The 2016 wildfire in Fort McMurray, Canada, intrigued us to study the functionality of the existing apps by analyzing social media information. We investigated a method to suggest features that are useful for emergency apps. Our proposed method called MAPFEAT, combines various machine learning techniques to analyze tweets in conjunction with crowdsourcing and guides an extended search in app stores to find currently missing features in emergency apps based on the needs stated in social media. MAPFEAT is evaluated by a real-world case study of the Fort McMurray wildfire, where we analyzed 69,680 unique tweets recorded over a period from May 2^{nd} to May 7^{th}, 2016.**

We found that (i) existing wildfire apps covered a range of 28 features with not all of them being considered helpful or essential, (ii) a large range of needs articulated in tweets can be mapped to features existing in non-emergency related apps, and (iii) MAPFEAT's suggested feature set is better aligned with the needs expressed by general public. Only six of the features existing in wildfire apps is among top 40 crowdsourced features explored by MAPFEAT, with the most important one just ranked 13^{th}. By using MAPFEAT, we proactively understand victims' needs and suggest mobile software support to the people impacted. MAPFEAT looks beyond the current functionality of apps in the same domain and extracts features using variety of crowdsourced data.

***Keywords*-Wildfire; Emergency; App store mining; Twitter; Mobile apps; Machine learning**

I. Introduction

There is evidence that social media platforms, especially Twitter, are widely used during emergency situations [13], [35]. In times like this, when staying connected is not a convenience but rather a necessity, mobile devices are a lifeline. During the 2016 Fort McMurray wildfire in Alberta, Canada, 80,000 people were evacuated and unable to return for over a month. As local radio stations went off the air and websites failed, social media became the crisis' unofficial emergency broadcast system.

Aware of the importance of mobile functionality during emergency situations, several applications were designed by NGOs (non government organizations), official state departments, and international humanitarian organizations. In particular, mobile apps for wildfire emergencies have been produced by the Red Cross and International Association of Fire Fighters, as well as the Canadian, United States, and Australian governments. However, our analysis of the apps available in Apple iTunes (Apple's app store) revealed that current wildfire app functionality is mainly limited to pushing notifications, sharing news, and providing access to fire maps. Our results show that the apps presently available to people in need are lacking relevant and useful features.

We investigate whether, and to what extend, existing wildfire apps reflect people's needs in emergency situations. We also suggest features to enhance these apps' functionality by harnessing the power of Twitter. To do so, we propose a method called MAPFEAT[1] and apply it to a case study of the Fort McMurray wildfire. MAPFEAT is a method for systematic tweet analysis and extended app store search. It explores software features that meet the needs of people in emergency situations.

MAPFEAT extracts needs from Twitter and maps them to features of apps already existing in the market (not exclusive to emergency apps). App stores provide a wealth of information about software features, as each app in app store is described by its technical functionality. MAPFEAT performs an extended search to support a better match between users' needs and actual functionality offered. This automated process and the use of crowdsourced data enables emergency app developers and owners to provide mobile software support that more effectively helps impacted people and facilitates the management of the crisis.

Suggesting functionality enhancements to emergency apps by mining tweets would primarily benefit ordinary people (also known as the general public) that were affected by the crisis. This support is intended to (i) facilitate the connection of impacted people to the emergency authorities and service providers, and (ii) enable people to connect to each other and help in possible cases. The method is evaluated by a case study where we report on the results of mining tweets published during the Fort McMurray wildfire.

In this paper, we answer the following research questions:

RQ1: How can tweets be mapped into mobile app features

[1]**M**ining **AP**p **FEA**tures from **T**weets: MAPFEAT

978-1-5386-2673-3/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSE-SEIS.2017.8

automatically?

Why and how? Past research has shown that mining social media services, particularly Twitter, is an effective way to manage emergency situations [12]. We expect that mining the needs of the general public and supporting them with software functionality would have a considerable impact on app design. We propose a method called MAPFEAT to perform this process systematically.

RQ2: For the Fort McMurray case study: How do features mined from MAPFEAT compare to the ones provided by existing wildfire apps?

Why and how? We compared the MAPFEAT results with features in existing wildfire apps by exploring which features in existing wildfire apps matched with needs stated in social media. We mined features from existing app descriptions following the method proposed by Harman et al. [10].

RQ3: For the Fort McMurray case study: How will the features generated by MAPFEAT be perceived by the general public?

Why and how? With the purpose of understanding the value of the MAPFEAT results, we crowdsourced a survey of 500 respondents. Respondents were given the list of features and asked to evaluate each feature as either essential, worthwhile, unimportant, or unwise.

To give an outline of the key idea and outcome of the paper, we provide a motivating example in the next section. In Section III, we offer a solution to **RQ1** by introducing MAPFEAT. Then, we describe the data set and design of our case study in Section IV. Using the case study, we answer **RQ2** in Section V. In section VI, we respond to **RQ3** by reporting the results of a survey of the general public. We conclude the paper by discussing threats to validity (Section VII), related work (Section VIII), and the relevancy and applicability of the results (Section IX).

II. Motivating example

We briefly illustrate the main idea of the paper with an example. Through this example, we provide a sneak peak into our method, MAPFEAT, and the results of its application. For the purpose of illustration, we selected a sample of tweets about the Fort McMurray wildfire, which can be seen in the text box below.

> May4 4:30pm: Moose Haven Lodge has rooms, food, gas avail. HWY 881 2 KM 217. Call 780-935-2372.
>
> Line for fuel - four pumps- snakes in Conklin.
>
> Any gas available between Fort Mac and Grasslands?
>
> YMMfire gasoline supplies to stations south along HWY 63 stable but lines ups from exodus causing some stations to not honour credit cards.
>
> Is there any gas at Wondering River?Line up?
>
> I'm in Boyle and have some gas, granola bars, and water if people are still stranded. Text (780)504-***8.

To automatically extract the topic of discussion from these tweets, we first lemmatized them by mapping the words into their dictionary form [15]. After applying topic modeling, we received the topic "gas, availability, lineup." Then, following the next step of MAPFEAT, we used this topic to run a thorough search in the app store and find all relevant apps. To search the app store, every combination of words in the topic with a length greater than one were sent to the Apple iTunes API:

(i) "gas" AND "availability" AND "lineup"

TABLE I
Features of the top ten apps retrieved by search queries from the Apple iTunes app store. The *italic* features were common to at least two apps retrieved by a search query.

(i) "gas" AND "availability" AND "lineup"		(ii) "gas" AND "availability"		(iii) "gas" AND "lineup"	
App	**Feature**	**App**	**Feature**	**App**	**Feature**
(i)-1	*Find cheapest gas based on your gas type*	(ii)-1	*Find cheapest gas* based on your gas type.	(iii)-1	Shows *gas prices* with easy to read maps.
(i)-2	*Share gas prices* across borders	(ii)-2	*Choose the nearest gas station*, bank and hospital based on your position.	(iii)-2	Crowd-sourced *gas prices* across borders
(i)-3	*Get the best Fuel Prices.*	(ii)-3	Gas delivered to your car - anytime, anywhere.	(iii)-3	Gas/diesel taxes 55 pieces of travel relevant information.
(i)-4	Gas/diesel taxes 55 pieces of travel relevant information for each of the 50 states.	(ii)-4	GPS station locator that provides real-time maps, directions. *Map a route to the selected station or enter a custom address.*	(iii)-4	Live routing based on community driven, *real-time traffic*. Find the *cheapest gas station* route with *lowest lineup*.
(i)-5	*Near real-time prices on your mobile device.* 10 Month Futures Chain. *Report gas prices*. More Crude Oil tracking: Sweet Light, Brent.	(ii)-5	*Find the cheapest gas* near. *Find gas stations* by distance/price. Report gas prices to help others find cheap gas and earn points. Filter stations by amenities and brands.	(iii)-5	Find route to gas station with *real time traffic, alerts and lineups*. Share location, route and ETA. Make a pit stop on your route and find the *cheapest gas* and other local amenities.

(ii) “gas" AND “availability"
(iii) “gas" AND “lineup"
(iv) “availability" AND “lineup"

We selected the top 10 apps retrieved by each search query, with the exception of (iv), which did not receive any results from Apple iTunes (none of the apps matched this search query). The features of the apps found by queries (i) to (iii) are presented in Table I. In some cases, separate search queries retrieved the same apps. Then, we collected and mined features from the app descriptions [10]. Some features are shared between different apps, such as (i)-4 and (iii)-3 in Table I. The features in italics are the ones common to at least two apps retrieved by a single search query. Only these features are of interest to the MAPFEAT process because they are mutually shared between apps, and thus, have a higher chance of matching the original topic. As a result of this process, we ended up with the following features:

- Find real-time gas prices
- Find cheapest gas
- Find nearest gas station
- Share gas prices
- View real-time traffic
- View lineups

In accordance with the last step of MAPFEAT, we finalized our mapping between tweets and app features by using crowdsourced evaluation to see if our extracted features were related to the tweet topic. We submitted the following question to `Amazon Mechanical Turk`[2], ensuring that exactly five workers answered it:

Select all features related to the topic "gas, availability, lineups":
☐ 1- Find real-time gas prices
☐ 2- Find cheapest gas
☐ 3- Find nearest gas station
☐ 4- Share gas prices
☐ 5- View real-time traffic
☐ 6- View lineups

The results of this crowd evaluation showed that all five of the workers agreed that the tweet topic “gas, availability, lineup" was related to features (1), (2), (3), (4), and (6). However, *four* of the workers believed that the topic is not related to feature (5), “view real-time traffic". Consequently, we mapped the tweet topic “gas, availability, lineups" to the following app features:

- Find real-time gas prices
- Find cheapest gas
- Find nearest gas station
- Share gas prices
- View lineups

Our study of the 26 existing wildfire apps showed that none of these features were currently implemented in any of these apps. However, people affected by such a crisis can benefit from having integrated information about the local price, availability, and convenience of gas stations within one app. Going a step further, we surveyed the crowd via `Amazon Mechanical Turk` and found that 53.2% of those surveyed consider these features to be essential in a wildfire app (average between five features).

So, while a feature like "view lineups" already exists in the app store as part of a navigation app, an emergency app with that feature would be better suited to address the needs of those involved in a wildfire emergency.

III. MAPFEAT: Mapping tweets to app features

The purpose of MAPFEAT is to introduce new features to emergency apps by offering a systematic method that transforms needs communicated over social media into software features. We propose a combination of techniques that automate the mining of needs from general purpose tweets and their translation into real app features using app store as the pool of possible features. The process of our proposed method is presented in Figure 1 and its steps are described in the following subsections.

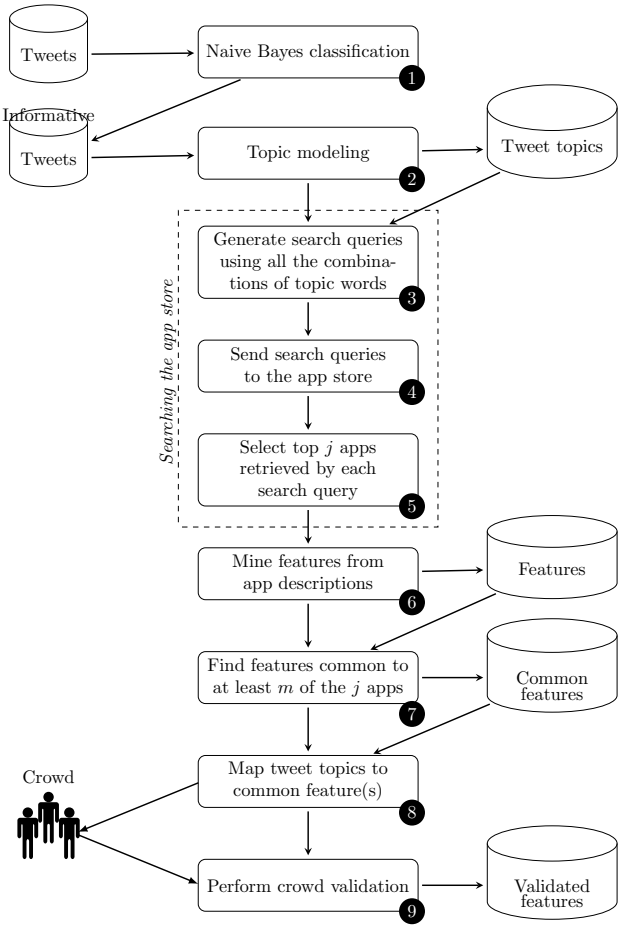

Fig. 1. MAPFEAT process of mapping tweets to app features

[2]https://www.mturk.com/mturk

A. Automatic classification of tweets

Focusing on a specific event gave us the chance to collect related tweets by following the trending keywords and hashtags of that event. This is represented by Step ❶ in Figure 1. Many tweets do not convey a need or requirement. Consider the following tweets:

"I don't care if you're right or wrong - seeing #ymmfire as an opportunity to talk politics or anything else is wrong."

"All my thoughts are with all the people who are affected by the devastation of the fire. #ymmfire #FortMacFire"

We defined the class of *non-informative tweets* as tweets lacking an explicit potential to be mapped into a software feature (like the above instances). *Informative tweets* are the ones that either express a requirement or offer support for fulfilling a requirement (see the tweets in the motivating example).

We used the Naive Bayes classifier, as it has been suggested by other studies for tweet classification [17]. Naive Bayes is based on the strong assumption that the existence of a word is independent of the existence of another word. For example, if the tweet contains the word "map", this gives us no further information about "traffic". The strength of Naive Bayes is that it is simple and has proved to perform well with a small training set [16].

B. Topic modeling

In the next step we used topic modeling to extract the in common topics between the needs stated in general purpose tweets (Step ❷ in Figure 1). The topic modeling of tweets was studied by Hong et al. [11]. LDA, known to work well on tweets [2], is an established method used to find common topics in a set of natural language text documents. Topics are created out of a combination of words that tend to appear together frequently in the documents of the corpus. LDA assumes there is a fixed number of *K* topics. It assigns a probability distribution over topics to each document. To decide which number of topics *K* made the most sense, two measures were suggested:

- the perplexity (log-likelihood of a held-out test set), where low perplexity indicates better generalizability of a topic [2], and
- intuitive meaningfulness of the results [5].

Each topic consists of a set (cluster) of w words that describe the topic. For example, if we decided to describe each topic by *three* words ($w = 3$), then LDA would group "gas", "traffic", and "map" together under one topic, and "accommodation", "room", and "hotel" under another topic.

C. Searching the app store

At this point, we have extracted topics from tweets and each topic is represented by a set of words. A software analyst can potentially map a tweet topic into software features. By systematically searching the app store, we not only automate this process, but also overcome the subjectivity involved in the human-based mapping process.

To this end, we generated multiple search queries using the combination of words in each topic (Step ❸ in Figure 1). We selected all combinations of words having a length ≥ 2 (Step ❹). Search queries with the $length = 1$ (Searching one word only) results in many general apps not really related to our purpose. We crawled the description of the top j apps retrieved by each search query (Step ❺).

D. Mining features from app description

Using the app descriptions gathered in the previous step, we then mined features (Step ❻ of Figure 1). To accomplish this, we followed the method proposed by Harman et al. [10], [27] to extract *featurelets* from app descriptions. They defined featurelets as "a set of commonly occurring co-located words, identified using NLTK's N-gram CollocationFinder package" [10], [27]. Following their proposed approach, we used a greedy hierarchical clustering to aggregate similar featurelets (having similarity $\geq 60\%$) [10]. As a result, we were able to extract features from the app descriptions, each consisting of two or three words (bitri-grams, i.e. 2-grams or 3-grams), which are considered to be the features provided by an app [27].

E. Mapping tweet topics to app features

So far, we described the process in which a tweet topic resulted in several search queries. Then, we extracted features from the description of the top apps retrieved for each search query. At this step, we mapped each search query to the features that were shared between at least m of the apps (Step ❼ in Figure 1). We did this because a specific keyword search retrieves a variety of apps, and thus, the commonality between their features is of special interest to us (we consider these functionalities as the reason that the apps were retrieved together by a specific search query). For the precise mapping, we present each tweet topic along with its corresponding features to the crowd and asked five workers to select all the features that are related to the tweet topic (see the motivation example). Considering the majority of votes, the mismatched features are excluded from the tweet topic.

Lastly, we mapped the original tweet topic to a set of features (Step ❽). The corresponding features of a tweet topic are defined as set of features received from every search query that originated from it. This way, we performed an extensive search of the app store and retrieved the union of all features found by search queries that originated from a specific tweet topic.

F. Evaluation

At this point, tweets have been mapped to app features. In order to validate the accuracy and value of the results, we performed crowdsourcing to evaluate the semantic relationship between each topic and its resulting features (Step ❾ in Figure 1). We automated this step by using the API of `Amazon Mechanical Turk`. We provided the extracted features

along with their corresponding tweet topic to the crowd and asked them to select all features relevant to the topic (see the motivating example). If the majority consider a feature not belonging to the topic, the respective feature is eliminated from the topic mapping list, otherwise there would be no change in the proposed feature set.

IV. Case Study Design

We recorded 69,680 unique tweets submitted in the context of the Fort McMurray wildfire over a period of May 2^nd to May 7^th, 2016. We applied MAPFEAT on these tweets as the proof of concept. In the following subsections, we describe the design of our case study. The results are presented in Sections V and VI (addressing RQ2 and RQ3, respectively).

A. Mining tweets about the Fort McMurray wildfire

Studying Twitter communication during emergency events is challenging. Access to the Twitter Search API is limited so we needed to decide what to collect at a stage when we only had limited information about the crisis.

To form the tweet database for our case study, we used the `Twitter Search API` to obtain publicly available tweets about Fort McMurray wildfire. The `Twitter search API` is restricted to 180 queries every 15 minutes and only returns tweets from the past 14 days. We used the trending hashtags #ymmfire, #FortMacfire, and #ymm to pull tweets connected to the Fort McMurray wildfire. We chose these hashtags through an initial investigation of the public Twitter stream.

We continuously gathered tweets related to these hashtags over the course of 6 days. The accumulation of this data resulted in 69,680 unique tweets.

Then, we randomly selected 2% of the tweets and performed manual analysis. First, we explored the usefulness of the data for our study. Second, we used this subset to train the Naive Bayes classifier to separate informative tweets from the non-informative ones.

B. Text pre-processing and lemmatization

Observing the common practice in text mining, we pre-processed tweets to make them ready for applying machine learning techniques such as Naive Bayes and LDA. We took the following steps:

1) Eliminated retweets,
2) Eliminated hashtags, emojis and special characters,
3) Eliminated URLs,
4) Eliminated duplicate tweets, and
5) Lemmatized tweets to map words into their dictionary form while retaining the context of the word [19], [15].

For pre-processing, we used an open source module called `Pattern` [30], which is built on top of the comprehensive `NLTK` python package [18]. We also used a python library, `Gensim`, to apply LDA and extract topics from the pre-processed tweets.

C. Survey design for evaluation of existing wildfire app features

All features were evaluated through crowdsourcing as the last step of MAPFEAT. We considered the crowd workers to be representative of the general public, with the assumption that they potentially might be involved in a fire emergency (often this probability provoke us to buy a fire insurance). We used a Kano-inspired survey [1] to evaluate the proposed features.

We asked the crowd to imagine that their hometown or place they were traveling was being destroyed by a wildfire and that they have their smart phone to help them in this situation. Then, we presented the features to them and asked:

> "*In your opinion, how important is it to have this feature as part of a wildfire emergency app?*"
>
> - It is essential
> - It is worthwhile
> - It is unimportant
> - It is unwise
> - I don't understand

To increase the validity of the results, we filtered out incomplete and low quality responses. A response was considered low quality if the respondent spent less than 20 seconds answering a question or selected the same answer for more than 90% of the questions. From the responses, we calculated the percentage of answers in each of the five categories for every feature. Then, we assigned the category with the highest percentage to the features and call it *an essential feature*, *a worthwhile feature*, *an unimportant features*, or *an unwise feature*. We used this type of survey to evaluate both the features in the existing apps and the features mined by MAPFEAT.

V. Comparison of features: generated by MAPFEAT vs. existing apps (RQ2)

We compared the features extracted by MAPFEAT with the ones in pre-existing apps for wildfire emergencies. For each feature, three scenarios were possible:

Scenario 1. The feature mined by MAPFEAT exists in the current wildfire apps.

Scenario 2. The feature mined by MAPFEAT does not exist in the current wildfire apps. In this case, MAPFEAT could provide a suggestion on a feature needed by general public. This can be taken by developers as a useful insight to design more desirable apps.

Scenario 3. The feature that exists in a current wildfire app was not mined by MAPFEAT. We interpret this either as a lack of justification for having such a feature, because it was not requested by the crowd, or incompleteness of the data. We suggest that app developers and software engineers inspect the usefulness of such features.

To answer **RQ2** and demonstrate the added value of our proposed method, we took three steps which are further described in the subsections below:

Step 1: Extract features from existing wildfire apps,

Step 2: Apply MAPFEAT to the Fort McMurray wildfire tweets, and
Step 3: Contrast the results of MAPFEAT with the existing app features.

A. Step 1: Extracting features from existing apps

In our study we focused on the Apple iTunes app store as it provides a free API for searching. We searched with “fire" and “wildfire" as keywords to find the apps available in the market. Our initial search resulted in 86 apps. We then read the app descriptions and filtered irrelevant apps from this set (for example, game and music player applications). We ended up with 26 apps that were related to wildfire emergencies.

Among the 26, one app was developed by the government of Alberta (the province in which Fort McMurray is located). This app was first released in 2013 and had three updates at the time this paper was written. Two other Canadian wildfires apps were found, one from British Colombia and the other from Saskatchewan, as well as three Australian apps from Tasmania, Victoria, and Queensland which all were developed by government and emergency department. Also, nine apps showed wildfire information for different cities of the United States, in addition to an official Red Cross app titled “Wildfires by American Red Cross." Lastly, there was an app from the International Association of Fire Fighters (IAFF). These 26 apps were published across seven different app store cate-

TABLE II
EVALUATION PROFILE OF TOP 40 RANKED FEATURES EXTRACTED BY MAPFEAT IN COMPARISON TO FEATURES OF EXISTING WILDFIRE APPS

Rank	**Feature**	**MAPFEAT**	**Baseline**	**% selected by survey participants**			
				Essential	Worthwhile	Unimportant	Unwise
R1	Fire alarm notification	✓	-	69.80%	22.04%	5.31%	1.63%
R2	Food and water requests and resources	✓	-	67.76%	22.04%	6.94%	2.86%
R3	Emergency maintenance service	✓	-	65.71%	24.49%	6.53%	2.04%
R4	Send emergency text messages	✓	-	65.31%	28.16%	4.90%	1.22%
R5	Safety guidelines	✓	-	64.98%	26.35%	6.14%	1.81%
R6	Fire and safeness warning	✓	-	64.90%	24.49%	8.16%	1.63%
R7	Request ambulance on a tap	✓	-	64.62%	22.74%	7.58%	4.33%
R8	Find nearest gas station	✓	-	63.90%	22.02%	8.30%	3.97%
R9	Emergency zones maps	✓	-	63.54%	25.27%	6.86%	3.97%
R10	Find a medical center	✓	-	61.01%	25.27%	10.11%	2.53%
R11	Subscribe for real time alerts	✓	-	60.82%	28.98%	6.94%	2.86%
R12	View gas lineups	✓	-	60.65%	26.71%	7.22%	4.69%
R13	Real-time fire information	✓	✓	60.55%	28.60%	6.31%	2.56%
R14	Fire education	✓	-	60.29%	26.71%	9.03%	3.25%
R15	Report incident	✓	-	60.29%	28.52%	7.22%	3.25%
R16	List of medical centers	✓	-	59.93%	27.80%	7.58%	3.61%
R17	Emergency kit list	✓	-	59.93%	28.88%	5.42%	4.69%
R18	Real time news notification	✓	✓	59.76%	28.60%	6.11%	4.54%
R19	Real time maps	✓	-	59.21%	25.99%	8.66%	4.69%
R20	Location-based fire information	✓	✓	58.78%	31.36%	5.72%	2.96%
R21	Gas availability & supply	✓	-	58.48%	28.16%	9.75%	3.61%
R22	List of fires	✓	-	57.76%	25.99%	11.19%	3.97%
R23	Direct police call	✓	-	57.04%	27.08%	10.47%	5.05%
R24	Instructions for before/during/after a wildfire	✓	✓	57.00%	31.95%	7.50%	2.17%
R25	Wildfire map	✓	✓	56.02%	31.76%	6.71%	3.75%
R26	Choose emergency types	✓	-	55.92%	34.69%	3.67%	3.27
R27	Real time update synchronization	✓	-	54.69%	32.24%	10.20%	1.22%
R28	Interactive learning procedure about fire	✓	-	52.71%	31.77%	9.03%	5.42%
R29	Device battery saving mode	✓	-	52.65%	31.84%	11.02%	4.08%
F30	Chat with aid respondents	✓	-	52.65%	36.73%	8.16%	2.45%
R31	Urgent health care provider request on tap	✓	-	51.99%	30.69%	13.36%	3.25%
R32	Patient transport request	✓	-	51.62%	33.21%	9.03%	5.05%
R33	Friends emergency contact	✓	-	50.90%	36.10%	8.30%	3.97%
R34	Live police chat	✓	-	50.54%	32.85%	10.11%	6.14%
R35	Up-to-date weather condition	✓	-	50.18%	32.85%	12.27%	3.97%
R36	Test fire vision distance and safety quality	✓	-	50.18%	33.21%	9.75%	5.42%
R37	Post a service request	✓	-	49.80%	35.51%	9.39%	2.86%
R38	Real-time news feed	✓	✓	49.70%	34.32%	11.24%	3.94%
R39	Evacuees residence and site list	✓	-	49.39%	30.20%	14.29	4.90%
R40	Emergency electricity maintenance request	✓	-	48.74%	35.38%	10.11%	5.05%

gories: lifestyle, news, weather, navigation, utilities, business, and education.

To extract features, we used a method proposed and evaluated by Harman et al. [10] as described in Section III-D. Using this method, we found a total of 28 unique app features distributed across all 26 apps.

B. Step 2: Applying MAPFEAT to Fort McMurray tweets

In this subsection, we describe the concrete data and results by applying the steps of MAPFEAT, as described above on Fort McMurray data. We first separated informative and non-informative tweets by applying the Naive Bayes classifier (Step ❶ in Figure 1). Applying 10-fold cross validation showed that the classifier has an accuracy of 75.8%, precision of 63.4%, and recall of 62.2%.

We then performed topic modeling on the informative tweets. Considering the perplexity and intuitiveness of the topics (see description of **RQ1**), we extracted 193 topics, each being described by *seven* words ($w = 7$). From using all the different combination of words (with combination of at least two words) as described in Step ❸ of Figure 1, we generated 23,400 search queries. We used the `Apple iTunes Search API` to retrieve apps for each search query. Each of these searches took 0.53 $seconds$ on average to get the results. We selected the top 10 apps from each search result ($j = 10$ in Step ❺ of Figure 1). 10,950 search queries did not return any apps. The results of all the search queries originating from a single topic were then aggregated. In total, our search queries resulted in 14,626 unique apps.

The motivation for conducting this extended search was to find software support (in form of apps' functionality) using the app store as a rich set of software features. For that purpose, we extracted features from all app descriptions that matched any of the search queries and selected features shared by at least two of the 10 apps retrieved per query ($m = 2$ in Step ❼ of Figure 1). To determine the similarity between two feature we set the threshold of cosine similarity measure to 60%. Doing this for all the retrieved apps of each search query, we found 188 features. We submitted these features along with their originating tweet topic into a crowd to verify the correct semantic relation between them. Crowd detected 13.2% semantic mismatch between the topics and extracted features. Excluding the mismatched feature and topics, we ended up in the final number of 163 features that were mapped into the tweets of Fort McMurray wildfire using MAPFEAT. The partial list of mined features is presented in Table II while the complete list of features is available online[3].

C. Step 3: Contrast MAPFEAT results with existing wildfire app features

MAPFEAT extracted 163 features considering the tweets about Fort McMurray wildfire. Having the features extracted by MAPFEAT, we compared them with features existing in the current wildfire apps. MAPFEAT could extract 139 features matching the needs of general public which were not considered within existing apps.

Our analysis showed that applying MAPFEAT on Fort McMurray tweets can find 85.7% of the features (i.e. 24 out of 28 features) available in existing wildfire apps. Using MAPFEAT, we mined 90% of the features that exist in more than one app. However, MAPFEAT did not mine the feature "follow location" that occurred twice in the existing wildfire apps. This might be because of data incompleteness or may indicate that general public does not need this feature(Scenario 3). The results of survey (**RQ3**) will shed some light on this by evaluating the importance of features for the crowd.

[3]http://www.ucalgary.ca/mnayebi/tools-and-data-sets

VI. RQ3: Evaluation of the wildfire app features

We performed another round of crowdsourced evaluation and asked the crowd how they perceive the value of the features mined by MAPFEAT. In other words, **RQ3** asks whether our proposed method generates app features that are deemed valuable by the general public in an emergency wildfire situation.

Having all the features extracted by MAPFEAT in addition to the *four* features in the existing apps that were not mined by MAPFEAT, we performed crowdsourcing. We asked 500 master (highly ranked) workers using `Amazon Mechanical Turk` to evaluate the importance of having each specific feature in the app. Using the results of the questionnaire (see Section IV-C), we provide in Table II the percentage of subjects who believe that a feature is essential, worthwhile, unimportant or unwise.

Scenario 1. **Features that were mined by MAPFEAT and currently exist in wildfire apps**

The results showed that the respondents found 25% of the 26 features of the existing wildfire apps as *essential*, while 35.7% of these features were evaluated as *worthwhile*. On the other side, 28.5% of the features were considered *unimportant* and 10.7% of them were evaluated as *unwise* ("flashlight" and "Generating and drawing wildfire maps" were considered unwise to be part of a wildfire app).

Comparing the crowd stated importance of these 24 features with the rest of the 139 features mined by MAPFEAT, shows the extensive mismatch between what the general public needs and what is currently provided in the wildfire apps. Looking into the results (partially reflected in Table II), shows the feature with highest "essential" degree that exists in the current wildfire apps, is ranked as #13. However, MAPFEAT could mine 12 features that had higher priority for public.

Scenario 2. **Features that were mined by MAPFEAT and do not exist in wildfire apps**

By crowdsourced evaluation of the 139 features mined by MAPFEAT, we found that:

- 28% of the features were classified as *essential*,
- 56.1% of the features were classified as *worthwhile*,
- 14.3% of the features were classified as *unimportant*, and

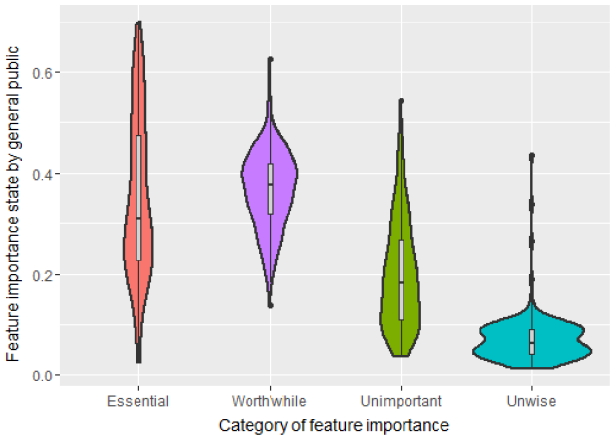

Fig. 2. Boxplots of crowdsourced feature evaluation

- 1.4% of the features were classified as *unwise*.

That means that about 84% of the features detected are worthwhile or even essential. The violin boxplots in Figure 2 show the frequency distribution for all four categories. For example, for the *worthwhile* category, the highest concentration is around 40%.

MAPFEAT introduced 139 features that were not available in the currently existing wildfire apps. These features are not new to the app market in general, however they were not offered as part of existing wildfire apps. MAPFEAT mined 39 essential features that not only covers all the already available features in the wildfire apps but also is 83.9% more features. In terms of quality, among the top 40 wildfire features mined from articulated needs, just six of them were present in the baseline feature set, with the most urgent one of them ranked 13^th.

Scenario 3. **Features that were not mined by MAPFEAT** Using MAPFEAT, we could not find one *unimportant feature* (51.08% of the respondents called it *unimportant*) and two unwise (39.05% and 42.41% called them unwise) features. So while MAPFEAT resulted in 144 essential and important features, it failed to find one worthwhile feature that already exists in the wildfire apps.

Three of the features that were missed by MAPFEAT were evaluated as *unimportant* and even *unwise*. This indicates that existence of some of the features in wildfire apps are not well justified as these were not communicated in the tweets nor being evaluated as *essential* or *worthwhile*. Also, MAPFEAT could not mine one worthwhile feature. This happened as we only used one data set for a specific wildfire. Other wildfire tweet data set, would likely lead us to explore more features for wildfire emergencies.

VII. Threats to Validity

We reported the results of a case study to show the process and effectiveness of MAPFEAT. In the following, we discuss threats to validity.

Construct validity: Considering tweets as a representative source for the needs of the general public might be of concerns for designing an app. However, we limited the application of MAPFEAT to emergency apps, as a huge body of research and analysis showed that tweet analysis is indeed an efficient aid to manage a wildfire.
We used crowdworkers for validation purposes. Another threat here is that subjects might not have first-hand experience with a wildfire emergency. As the context of the questions is easy to understand and fire protection is part of insurance plans and education, we consider the crowd as representative respondents. In addition, we had a substantial number of responses, from people directly impacted by the Fort McMurray emergency.

Conclusion validity: The number of crowd workers (500) we used is considered large enough to exclude randomness. The reliability of the crowd responses is considered high as we excluded responses made in less than 20 seconds and eliminated subjects selecting the same answer for more than 90% of the questions.

Internal validity: By searching for specific hashtags, we have ensured that all tweets are connected to the Fort McMurray wildfire. We chose these hashtags through an initial investigation of this emergency.
Features extracted by MAPFEAT and features from existing apps can be imprecise as this is largely investigated in app store mining research. However, we selected the method [10] that were checked for sanity in several research studies. We would also argue that the crowdsourced evaluation mitigates the influence of this phenomenon.
MAPFEAT does not claim to find all features necessary in an emergency app. The completeness is dependent on data as well as the process. A needed feature in a wildfire emergency might not have been communicated in the tweets of Fort McMurray. Also, during the nine step process, there are several assumptions made about parameters. The number of words used for defining a topic could be done in a different way which potentially enlarges the search space. However, we do not argue for completeness, but just for correctness and added value. We also used crowdsourcing to back up the process validation.

External validity: We evaluated MAPFEAT through a comprehensive case study. From there, we do not claim generalization of results to other social media or to other events. We argue that the application of MAPFEAT could provide similar support in other situations as long as the tweet data set is representative of that event. Use of tweets for managing emergency events that involve masses of people were proved to be informative.

VIII. Related Work

A. Twitter analysis for emergency response

Social media services like Twitter have emerged as a popular medium for communication. Twitter data offers a rich mechanism for exploring events before, during, and after emergen-

cies. Before an emergency, data can be monitored and analyzed to identify events [6]. Various research studies examined the use of Twitter data during emergency events [12]. Vieweg et al. [33] analyzed Twitter communication during the 2009 Red River Floods and Oklahoma grass fires. Hughes et al. [14] also analyzed online public communication with the police and fire departments during hurricane Sandy in USA. The study of Takahashi et al. [31] during Typhoon Haiyan in Philippines focused on the general public and organizational usage of Twitter. Also, Power et al. [26] developed a notification system to identify fire events in Australia.

Different data mining techniques, such as anomaly detection by clustering [32], [4] and topic modeling using textual data [34], [36], have been applied to tweets in order to bring insight during emergency events. Using classifiers to separate informative from non-informative tweets is also established in this context [23]. The use of Twitter data for managing emergencies is evolving rapidly for different types of events [12]. However, to the best of our knowledge this paper is the first study that focuses on requirements elicitation of Twitter data for mobile app mining.

B. Mining software features from App stores and product descriptions

App store mining and analysis has been widely studied in the context of software engineering [22]. Mining features from apps has been studied by analyzing the app description [10], [8], as described in this paper, or by mining app reviews [19]. Both approaches were proven to extract comprehensive yet incomplete sets of app features, as neither the description nor reviews mention all features. Although, knowing this to be a threat, many research studies base their analysis and decision-making for app development on the features being mined from the app descriptions [22]. Looking into app features as a source for developing new ideas was also discussed by Maalej et al. [20].

On the other hand, mining the public description of software products to reduce effort and help the domain analysis process for tradition software products, such as Anti-Virus products has been discussed [9]. In the same context, a recommendation system for modeling product features in a specific domain have been designed and proposed [7]. MAPFEAT differs from these methods as it does not necessarily look into a specific domain, rather, it globally searches the app store.

C. Crowdsourcing in Software Engineering

The wisdom of the crowd is increasingly used across different disciplines of software engineering [21]. Crowdsourcing for the purpose of requirements elicitation and categorization has been used [29], and tweets themselves are a form of crowdsourced data [3]. We used tweet data from the crowd to elicit needs and map them into software requirements, as well as, verifying the mapping between tweet topics and the retrieved corresponding features. Crowdsourced human-assisted verification was first studied by Li et al. [24]. We also used crowdsourcing to validate the importance of features. Sustainability of crowdsourcing for software products was studied by [25] and mechanisms to enhance sustainability of crowsourced software were suggested in [25], [28].

IX. Relevance and Applicability

Information about societal problems is typically distributed over different locations and systems. This is particularly true for information needed in various kinds of emergency situations. The challenge tackled by this paper is to define and execute a systematic process to combine different sources of data. Intrigued by the initial evidence that existing apps for handling wildfire emergency situations were lacking valuable functionality, we designed a process looking beyond contextual boundaries and searched the app store holistically for the user needs.

The systematic method of understanding user needs and searching them to map it to the existing features of non-emergency app products has the potential for application beyond wildfire. The application of our proposed method relies on the:

- Mass impact of the event,
- Wide usage of mobile apps in the event context, and
- Usage of social media to stay connected, communicate, and share experience and opinions about the event.

The above conditions are prerequisite for applying MAPFEAT and are all satisfied in mass emergencies such as earthquakes, flooding and wildfires. Another common feature of emergency situations is the need to explore a wide range of dimensions by thinking “outside the box." This is difficult to accomplish solely in a proactive manner. Yet, MAPFEAT is not even limited to emergency situations and can be used for finding new features to provide better software support in day to day life of people addressing scenarios in the economic, political, environmental, social and technical aspects of society and urban life.

X. Conclusions

MAPFEAT is an automated method designed for the purpose of mapping general public needs that were communicated via Twitter into mobile app features. It uniquely combines different forms of crowdsourcing with existing techniques for classification, natural language processing, and topic modeling. The key idea is to map topics extracted from a set of event-related tweets to features that already exist in the apps apart from the app category or scope of functionality. This way, MAPFEAT enhances the design of an app by transferring people’s stated needs in social media into technological functionality.

The feasibility of the method was demonstrated by a real-world case study. Based on the textual analysis of 69,680 unique tweets, we could specify a significant amount of new functionality (139 new features) demonstrated to be valuable for general public users in wildfire emergency events. The main conclusion of the case study is that MAPFEAT is able to find a significant number of additional features which

were not in the currently existing wildfire apps. 87.1% of these features were confirmed useful by general public. The additional features that we elicited, was evaluated as clearly more comprehensive and more helpful functionality comparing to what was offered in the existing apps. For app providers, these findings are considered of substantial value to better support victims in future emergency events.

ACKNOWLEDGMENTS

This research was partially supported by the Natural Sciences and Engineering Research Council of Canada, NSERC Discovery Grant 250343-12 and Alberta Innovates Technology Futures.

REFERENCES

[1] A. Begel and T. Zimmermann. Analyze this! 145 questions for data scientists in software engineering. In *Proceedings of the 36th International Conference on Software Engineering*, pages 12–23. ACM, 2014.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[3] D. Boyd, S. Golder, and G. Lotan. Tweet, tweet, retweet: Conversational aspects of retweeting on twitter. In *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, pages 1–10. IEEE, 2010.

[4] J. Chae, D. Thom, H. Bosch, Y. Jang, R. Maciejewski, D. S. Ebert, and T. Ertl. Spatiotemporal social media analytics for abnormal event detection and examination using seasonal-trend decomposition. In *IEEE Conference on Visual Analytics Science and Technology 2012, VAST 2012 - Proceedings*, pages 143–152.

[5] N. Chen, J. Lin, S. C. Hoi, X. Xiao, and B. Zhang. Ar-miner: mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th International Conference on Software Engineering*, pages 767–778. ACM, 2014.

[6] W. Dou, X. Wang, D. Skau, W. Ribarsky, and M. X. Zhou. Leadline: Interactive visual analysis of text data through event identification and exploration. In *IEEE Conference on Visual Analytics Science and Technology 2012, VAST 2012 - Proceedings*, pages 93–102.

[7] H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B. Mobasher, C. Castro-Herrera, and M. Mirakhorli. On-demand feature recommendations derived from mining public product descriptions. In *2011 33rd International Conference on Software Engineering (ICSE)*, pages 181–190. IEEE, 2011.

[8] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller. Checking app behavior against app descriptions. In *Proceedings of the 36th International Conference on Software Engineering*, pages 1025–1035. ACM, 2014.

[9] N. Hariri, C. Castro-Herrera, M. Mirakhorli, J. Cleland-Huang, and B. Mobasher. Supporting domain analysis through mining and recommending features from online product listings. *IEEE Transactions on Software Engineering*, 39(12):1736–1752, 2013.

[10] M. Harman, Y. Jia, and Y. Zhang. App store mining and analysis: Msr for app stores. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*, pages 108–111. IEEE Press, 2012.

[11] L. Hong and B. D. Davison. Empirical study of topic modeling in twitter. In *Proceedings of the first workshop on social media analytics*, pages 80–88. ACM, 2010.

[12] J. B. Houston, J. Hawthorne, M. F. Perreault, E. H. Park, M. Goldstein Hode, M. R. Halliwell, S. E. Turner McGowen, R. Davis, S. Vaid, J. A. McElderry, and S. A. Griffith. Social media and disasters: a functional framework for social media use in disaster planning, response, and research. *Disasters*, 39(1):1–22, 2015.

[13] A. L. Hughes and L. Palen. Twitter adoption and use in mass convergence and emergency events. *International Journal of Emergency Management*, 6(3-4):248–260, 2009.

[14] A. L. Hughes, L. A. A. St Denis, L. Palen, and K. M. Anderson. Online public communications by police fire services during the 2012 hurricane sandy. pages 1505–1514

[15] A. G. Jivani et al. A comparative study of stemming algorithms. *Int. J. Comp. Tech. Appl*, 2(6):1930–1938, 2011.

[16] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, pages 3–24, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press.

[17] K. Lee, D. Palsetia, R. Narayanan, M. M. A. Patwary, A. Agrawal, and A. Choudhary. Twitter trending topic classification. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 251–258. IEEE, 2011.

[18] E. Loper and S. Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1*, pages 63–70. Association for Computational Linguistics, 2002.

[19] W. Maalej and H. Nabil. Bug report, feature request, or simply praise? on automatically classifying app reviews. In *2015 IEEE 23rd international requirements engineering conference (RE)*, pages 116–125. IEEE, 2015.

[20] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe. Toward data-driven requirements engineering. *Software, IEEE*, 33(1):48–54, 2016.

[21] K. Mao, L. Capra, M. Harman, and Y. Jia. A survey of the use of crowdsourcing in software engineering. *RN*, 15(01), 2015.

[22] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman. A survey of app store analysis for software engineering. *RN*, 16:02, 2016.

[23] S. P. Moon, Y. Liu, S. Entezari, A. Pirzadeh, A. Pappas, and M. S. Pfaff. Top health trends: An information visualization tool for awareness of local health trends. In *ISCRAM 2013 Conference Proceedings - 10th International Conference on Information Systems for Crisis Response and Management*, pages 177–187.

[24] R. Musson, J. Richards, D. Fisher, C. Bird, B. Bussone, and S. Ganguly. Leveraging the crowd: how 48,000 users helped improve lync performance. *IEEE software*, 30(4):38–45, 2013.

[25] D. Pilz and H. Gewald. Does money matter? motivational factors for participation in paid-and non-profit-crowdsourcing communities. In *Wirtschaftsinformatik*, page 37, 2013.

[26] R. Power, B. Robinson, and D. Ratcliffe. Finding fires with twitter. In *Proceedings of the Australasian Language Technology Association (ALTA) Workshop, Brisbane, Australia*, pages 80–89.

[27] F. Sarro, A. A. Al-Subaihin, M. Harman, Y. Jia, W. Martin, and Y. Zhang. Feature lifecycles as they spread, migrate, remain, and die in app stores. In *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, pages 76–85. IEEE, 2015.

[28] E. Schenk and C. Guittard. Crowdsourcing: What can be outsourced to the crowd, and why. In *Workshop on Open Source Innovation, Strasbourg, France*, page 72, 2009.

[29] N. Seyff, F. Graf, and N. Maiden. Using mobile re tools to give end-users their own voice. In *2010 18th IEEE International Requirements Engineering Conference*, pages 37–46. IEEE, 2010.

[30] T. D. Smedt and W. Daelemans. Pattern for python. *Journal of Machine Learning Research*, 13(Jun):2063–2067, 2012.

[31] B. Takahashi, E. C. Tandoc, and C. Carmichael. Communicating on twitter during a disaster: An analysis of tweets during typhoon haiyan in the philippines. *Computers in Human Behavior*, 50:392–398 Elsevier.

[32] D. Thom, H. Bosch, S. Koch, M. Wörner, and T. Ertl. Spatiotemporal anomaly detection through visual analysis of geolocated twitter messages. *Pacific Visualization 2012*, 2012.

[33] S. Vieweg, A. L. Hughes, K. Starbird, and L. Palen. Microblogging during two natural hazards events: What twitter may contribute to situational awareness. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1079–1088, New York, NY, USA, 2010. ACM.

[34] S. Yang, H. Chung, X. Lin, S. Lee, L. Chen, A. Wood, A. L. Kavanaugh, S. D. Sheetz, D. J. Shoemaker, and E. A. Fox. Phasevis1: What, when, where, and who in visualizing the four phases of emergency management through the lens of social media. In *ISCRAM 2013 Conference Proceedings - 10th International Conference on Information Systems for Crisis Response and Management*, pages 912–917.

[35] J. Yin, A. Lampert, M. Cameron, B. Robinson, and R. Power. Using social media to enhance emergency situation awareness. *IEEE Intelligent Systems*, 27(6):52–59, 2012.

[36] D. Zhang and Z. Xie. Analysis and research on microblogging network model based on crawler data. In *Proceedings of 2011 International Conference on Computer Science and Network Technology, ICCSNT 2011*, volume 2, pages 653–656.