Smarter Crisis Crowdsourcing

Kayla Jacobs
Computer Science
Technion
Haifa, Israel 3200003
kayla@cs.technion.ac.il

Kwang-Sung Jun
Dept. of Computer Sciences
U. of Wisconsin-Madison
Madison, WI 53706
deltakam@cs.wisc.edu

Nathan Leiby
Clever
San Francisco, CA 94105
nathan.leiby@clever.com

Elena Eneva
Accenture
San Francisco, CA 94105
elena@elenaeneva.com

ABSTRACT

In crisis situations like contested elections, natural disasters, and humanitarian abuses, crowdsourcing helps gather, review, and map incident reports. This aids communication between victims/civilians and responders such as election monitors, relief agencies, and journalists. However, processing these reports currently relies on human reviewers for manual anonymization, categorization, and geo-location—which is slow, tedious, and difficult to scale up in fast-paced or high-volume situations.

With our partner Ushahidi, we developed Ushine Learning, open-source plug-in tools to assist reviewers by providing automated suggestions for common tasks like categorizing report topics, flagging private info for anonymizing, identifying locations and URLs, determining report language, and detecting duplicate reports to minimize redundancy.

To evaluate our system, we measured human reviewers' speed and accuracy in a simulated contested election crisis. The results showed our system slightly improves the performance of reviewers, and that higher-quality suggestions can help even further, motivating system improvement.

We conclude with future directions and ethical considerations for automated support of crowdsourced crisis review.

1. INTRODUCTION

1.1 Crisis Crowdsourcing

In 2008 in Kenya, contested elections sparked widespread instability and violence. In 2010 in Pakistan, massive flooding displaced millions. Today in Syria, civil war has created a humanitarian disaster for civilians. In these kinds of crisis situations, it's hard to know what's happening. There's an information gap between the information providers (voters, natural disaster survivors, and victims) and the information responders (election monitors, aid organizations, activists, NGOs and journalists).

Crowdsourced crisis reporting platforms—including those developed by our partner, Kenyan non-profit Ushahidi [9]—aim to narrow this information gap. They provide centralized software to collect, curate, and publish reports coming from the ground. Figure 1 summarizes the typical progression of a report.

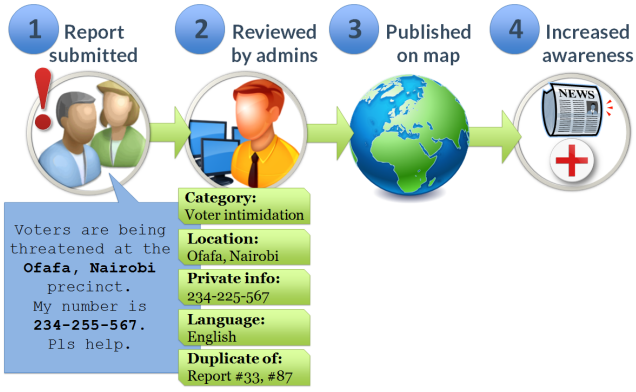

Figure 1: The crisis crowdsourcing process.

1. **Report submitted:** A citizen or victim submits a message via SMS,[1] e-mail, web form, smartphone application, or Twitter.
2. **Reviewed by admins:** Human reviewers (often volunteers or response organization administrators) assess the report. They identify the text's language and determine whether translation is needed or if the report needs to be routed to another volunteer. They ensure that the report is not a duplicate of an existing message, and remove any sensitive information, like telephone numbers and names, that could put people at risk. Finally, they geo-locate and categorize the message content.
3. **Published on map:** After review and possible editing, the report is published on an easily-readable map.

[1]In developing regions, or when natural disasters or unrest those with damaged infrastructure due to natural disaster or unrest, SMS is often the most-available and most-reliable means of communication. However, in sensitive political situations, it suffers from data security limitations.

KDD 2014 New York, NY USA

4. **Increased awareness:** The crisis situation benefits from increased awareness, and responders like aid organizations or journalists can take better-informed actions to help.

Crisis crowdsourcing has already been used in hundreds of situations, including election monitoring in Kenya [10], natural disaster damage assessment during Hurricane Sandy [5] and Haiti [6] (thoroughly analyzed in [14]), monitoring human rights and security in Nepal [7], and tracking violence in Syria during the civil war [8]. Similar technologies have also been used for non-crisis situations like repairing city roads in Ireland [2], and even humorously tracking "zombie sightings" worldwide [11].

The technical platform, duration, scale, and goal of each effort can vary considerably, but they all share a common focus on crowdsourcing information to better understand, and respond to, a challenging situation.

1.2 The Problem

Currently, the review process (step 2 in Figure 1) is heavily manual and labor-intensive. It's slow, tedious, often requires domain experts, and doesn't scale up well for large volumes (which may overwhelm reviewers) or fast-paced situations (where quick turn-around time is critical).

For example, during the 2010 earthquake in Haiti, thousands of Haitian Kreyol and French-speaking volunteers from dozens of countries collaborated to process 80,000 short SMS reports. While the average review time during the crisis was an impressive five minutes per report, the collective efforts were extremely intensive and impossible without the large, motivated volunteer community focused on the time-critical nature of the processing [14]. On the opposite end of the spectrum, a humanitarian abuse tracking project described their review scope as several dozen long reports per week, managed—with difficulty—by two part-time journalist reviewers.

Additionally, reports often contain sensitive or private information. The people who submit—or are mentioned in—reports may be especially vulnerable, including children or civilians in unstable settings. While anonymization is typically a part of manual report reviewing, it is too easy to miss a piece of sensitive information.

1.3 Our Solution

Using natural language processing and machine learning, we developed Ushine Learning, a suite of free, open-source tools that support and improve the human review process. With these tools, crisis report reviewers can reduce the time and tedium associated with processing reports, and focus their energies on verifying and responding to the reports instead.

2. USHINE LEARNING

2.1 Tasks

After careful consultation with Ushahidi experts and leaders from several worldwide crisis responder networks, we identified six common reviewer tasks where automated suggestions are needed, and built tools to help:

1. **Detect near-duplicate reports:** Has this report already been submitted? Often the same report is submitted multiple times, by accident or because the sender wasn't sure of receipt. Near-duplicates are also common when the report source is Twitter, due to retweets. Reviewing redundant reports is time-consuming.
2. **Detect language:** What language(s) must the reviewer know? Is translation needed? Often, reports are submitted in one of several local languages.
3. **Identify personally identifiable information (PII):** Does the report contain any names, addresses, phone or ID numbers, etc. that need anonymizing before sharing or publishing?
4. **Identify locations:** Where to geo-locate the report? Extracting addresses or other location information from the report is needed to place it on a map.
5. **Identify URLs (links):** Reports often contain relevant URLs to photos, news articles, videos, etc. that serve as information sources and/or validation references. It is time-consuming and tedious for reviewers to pull out the links.
6. **Categorize report:** Is this report describing bribery or violence, food shortages or illness? Reviewers assign non-exclusive topic categories to each report for later filtering and analysis.

2.2 System Architecture

Ushine Learning's system architecture consists of a Machine Learning Module written in Python, wrapped by a Flask Webapp which communicates with an Ushahidi plug-in using a REST-ful API and JSON objects. Our plug-in's data synchronization and UI integration required core changes in the Ushahidi platform, as well as the plug-in. These changes were made for Ushahidi 2.x, and we are exploring incorporation into Ushahidi 3.0. While we developed a plug-in for Ushahidi's platform, our system can be easily adapted to interact with other platforms as well.

The crisis crowdsourcing platform (Ushahidi or otherwise) receives automated suggestions from our system and can act on them in whatever way is most appropriate. Some of our tools, like URL and language detection, are robust enough that they generally can be automatically acted upon safely (perhaps with a built-in option for humans to override the automatic classifications if they disagree with the system's suggestions). Other tasks that our tools address, like category suggestions, location identification, and especially PII detection, are more subjective and potentially sensitive undertakings that we warn can be too high-stakes to entrust to our system alone. We advocate that in such cases, Ushine Learning's information should be presented to the reviewers as suggestions to help them make their review decisions (for example, as colored highlighting), but should not replace human thoughtfulness and decision-making. This is especially important for the PII identification tool, as will be discussed further in Section 4.3.

2.3 Data Challenges

Several challenges appeared frequently in the crisis datasets we examined and used for training and evaluation of our tools. Many reports contained extensive use of abbreviations, "SMS-speak" (the writing style associated with cell-phone text messaging), misspellings, and relatively short

report lengths. In some datasets there were multiple languages, sometimes even within the same report. Naturally, these challenges affected the scope of natural language processing and machine learning techniques on the report text, and shaped our choices of approaches. It also led us to focus, for now, on reports written in English, for which natural language processing resources are most mature.

2.4 Tool Implementations

- **Detect near-duplicate reports:** We defined duplicate reports as those separated by short distances with respect to their efficiently-calculated SimHash [15] representations. Based on a manual analysis of 500 reports, we found that a similarity score threshold of 0.875 gave the best results for English reports. We expect this tool to be language independent, although further study is needed of how the threshold might change for different languages.

- **Detect language:** We used the `langid` Python package [13]. When tested on 850 reports in various language for agreement with manual language identification, this plug-in performed well with 85% accuracy, beating two other tested Python packages (`cld` [1] and `guess_language` [4]). While Google Translate [3] performed better, with 87% accuracy, it was not a free resource, a severe limitation for its intended application.

- **Identify PII, locations, and URLs:** We worked with a pre-trained named entity recognizer (NER) from the NLTK [12] Python package to identify names and locations. For non-English languages, a different NER, trained on that language, would need to be used instead. We used regular expressions to detect e-mail addresses, phone/ID numbers, and URLs, which are also mostly language-independent (in the case of phone numbers, a region-specific telephone regular expression is required). In general, for PII, we chose to be overly careful: false negatives are more dangerous than false positives. (In other words, we favored recall over precision.)

- **Categorize report:** We trained support vector machine (SVM) classifiers on thousands of manually categorized reports from past crises. For classification features, we used bag-of-words unigram frequencies with a cut-off of 5; we found unigram TF-IDF and bigram frequencies were unhelpful. A major challenge was the "cold start problem:" a classifier trained only on the current situation's existing reports has dismal performance at first, when training data is scarce, but improves over time by learning from incoming reports. Conversely, a classifier pre-trained on similar crisis data (e.g., past contested elections) performs well at first but doesn't improve or learn from incoming reports. We combined the two types of classifiers to create an adaptive classifier—one that starts off with pre-trained data yet learns from incoming reports—that performs at least as well as both at every point of the situation.

Note for all tasks besides category suggestions (for which we built our own classifiers) we adapted existing, free, open-source natural language processing tools.

3. EXTERNAL EVALUATION EXPERIMENT

To evaluate the practical usefulness of automatic suggestions to report reviewers, we built an interactive website to simulate a crisis situation based an 2010 election monitoring effort in the Philippines. The experiment tested the speed and accuracy of ∼200 reviewers completing four tasks for each of ∼25 report: identifying private information, geographical locations, and URLs within the text; and report categorization.

3.1 The Three Experiment Interfaces

Each reviewer annotated the same series of reports in the same order, but upon beginning the experiment was randomly assigned to one of three system interfaces (see Figure 2):

1. **Plain:** completely manual interface with no review suggestions, approximating the existing report review interface in Ushahidi (our baseline).

2. **Machine**: enhanced interface with our system's (possibly incorrect) suggestions. Personal information, locations, and URLs were highlighted in different colors, and graphical scores rated the report's relevance for each category.

3. **Oracle**: identical to the Machine interface, except that all the suggestions were secretly hand-crafted to be perfectly correct.

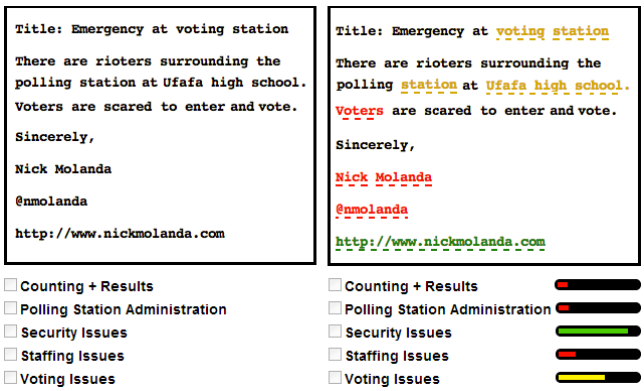

Figure 2: Plain interface (left) and Machine interface (right). The Oracle interface is nearly identical to the latter, except with only *correct* suggestions.

Comparing average user performance on the Machine and Oracle versions addresses the question of "Are our system's suggestions good?" The larger the gap, the more room our tools have to improve.

Comparing average user performance on the Plain and Oracle versions addresses the more fundamental question of "how helpful are good suggestions to the report reviewer?" The larger the gap, the greater the potential impact of a high-quality suggestion systems to help report reviewers.

We expected that Oracle would outperform Plain, and that Machine would be somewhere in the middle. But *where* in the middle — far from Oracle (meaning our system performed poorly) or close (meaning it performed well)? We also considered the uncomfortable possibility that even the Oracle might not necessarily be a lot better than Plain—

and that it conceivably might even be worse, if the presence of highlighted suggestions somehow confused people or slowed them down.

3.2 Results

Table 1 summarizes the average time for each review task. Overall, the Oracle interface saved 5 seconds on average compared to the Plain interface, a statistically-significant ∼10% improvement (using Welch's t test [16] with a 95% confidence interval), while the improvement from the Machine version was not statistically significant. We conclude that automatic suggestions help make reviewing faster as suggestions become more accurate, and that our Ushine Learning's suggestions require improvement.

	Plain	**Machine**	**Oracle**
PII	18.57	18.90	16.26
Location	12.56	11.90	11.16
URL	6.45	6.48	5.40
Categorization	13.74	13.46	13.39
Average	51.31	50.75	46.21

Table 1: Average reviewer time (in seconds).

We were surprised that the Machine interface was slower than the Oracle one for the URL identification task, given that URL detection through regular expressions is easy. We reexamined our code and discovered a small and easily-fixed bug—making our original suggestions faulty and explaining their poor performance.

The Oracle interface often outperformed Machine because in the latter there were sometimes erroneous suggestions; we hypothesize that incorrect suggestions distracted reviewers and caused hesitations.

Table 2 measures the interface impact on reviewer correctness by computing a balanced F1 score to summarize precision/recall for each task. Overall, the interfaces had similar correctness measurements, with the exception of a significant improvement in categorization correctness in the Oracle interface compared to the other two.

	Plain	**Machine**	**Oracle**
PII	73%	73%	75%
Location	94%	97%	96%
URL	97%	96%	95%
Categorization	77%	75%	84%

Table 2: Average reviewer accuracy (percent of choices correct).

4. CONCLUSIONS

4.1 Summary

We developed Ushine Learning, a suite of free, open-source, plug-in tools to assist human reviewers of crowdsourced crisis reports. Our system provides automated suggestions for six common reviewer tasks (detecting near-duplicate reports; detecting report language; categorizing the report; and identifying personally identifiable information, locations, and URLs in the report text).

Several of the tasks were evaluated through an external evaluation experiment testing the contribution of our system's suggestions to human reviewer speed and accuracy. Ushine Learning unfortunately did not result in statistically-significant, though it did not worsen matters either. However, the "perfect suggestions" Oracle interface showed a significant speed-ups in most tasks, and accuracy improvement in the categorization task. This motivates further improvements to our system, and gives confidence that the project goals are worthwhile and useful in the real world.

4.2 Future Work

- Automatic support for additional common reviewer tasks identified by the crisis crowdsourcing community, such as: detecting irrelevant reports, estimating report urgency, clustering similar (though non-identical) reports, sentiment analysis of reports, and recommending new or consolidated categories.
- Exploring the impact of automatic suggestion presentation on reviewer behavior. Is highlighting the optimal way of showing suggestions? How (if at all) should suggestion confidence scores be displayed?
- Integration with other reviewer support paradigms besides those relying on automatic suggestions (e.g., George Chamales's micro-tasking framework).
- Expansion of tools to non-English languages.
- More extensive software integration with newer versions of Ushahidi, and with other crowdsourced crisis technologies.
- Analysis of reactions to Ushine Learning from real-life (as opposed to simulated) crisis report reviewers.

In retrospect, before beginning development work we ought to have first run the evaluation experiment with just the Plain and Oracle interfaces (by definition, the Machine interface wouldn't have existed yet) to ensure there existed a sufficiently large gap—as was later confirmed. In the future, new proposed tasks should follow this plan, to ensure tool usefulness is demonstrably real rather than just assumed.

4.3 Ethical Considerations

- The definition of "sensitive" information in text requiring removal/anonymization is a subtle and context-specific one, and our tool–while meant to aid reviewers—is not perfect. This tool should never replace, but rather should support, thoughtful human decision-making. (Of course, humans are not perfectly accurate, either!)
- Some tools, especially category classification, need access to training data from prior crisis reports to improve. Who should have access? How should they be safely stored and shared? How is informed permission obtained? How are these privacy concerns balanced with the goal of improving Ushine Learning?
- Even if a report submitter or an administrator agrees to share their data, he or she may not fully understand the potential consequences—intentional or not—of providing the information. Care should be taken to ensure that sharing permission is as truly informed and safe as possible.

Acknowledgments

This work was done as part of the Eric and Wendy Schmidt Data Science for Social Good Fellowship. Hearty thanks to our partners at Ushahidi and the many individuals and organizations who generously gave us their advice, insights and feedback, including (alphabetically) Chris Albon, Rob Baker, Colm Bradley, George Chamales, Jennifer Chan, Crisis Mappers, Schuyler Erle, Sara-Jayne Farmer, Jordan Freud, Rayid Ghani, Eric Goodwin, Catherine Graham, Neil Horning, Humanity Road, Anahi Ayala Iacucci, Emmanuel Kala, David Kobia, Heather Leson, Nick Mader, Rob Mitchum, Rob Munro, Angela Oduor, Hunter Owens, Emily Rowe, Syria Tracker, Chris Thompson, and Juan-Pablo Velez.

5. REFERENCES

[1] cld Python package. `https://github.com/mzsanford/cld`.

[2] Fix Your Street. `http://www.fixyourstreet.ie`.

[3] Google Translate. `http://translate.google.com`.

[4] guess-language Python package. `https://pypi.python.org/pypi/guess-language`.

[5] Huffington Post Sandy Stormwatch. `https://hpsandy.crowdmap.com`.

[6] Mission 4636. `http://www.mission4636.org`.

[7] Nepal Monitor. `https://www.nepalmonitor.org`.

[8] Syria Tracker. `https://syriatracker.crowdmap.com`.

[9] Ushahidi. `http://ushahidi.com/`.

[10] Ushahidi (Legacy). `http://legacy.ushahidi.com`.

[11] Zombie Reports. `http://zombiereports.com`.

[12] 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*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[13] M. Lui. Language identification (LangID) Python package. `https://github.com/saffsd/langid.py`.

[14] R. Munro. Crowdsourcing and the crisis-affected community: Lessons learned and looking forward from Mission 4636. *Inf. Retr.*, 16(2):210–266, 2013.

[15] C. Sadowski and G. Levin. Simhash: Hash-based similarity detection. Technical report, Google, 2007.

[16] B. L. Welch. The Generalization of 'Student's' Problem when Several Different Population Variances are Involved. *Biometrika*, (1/2):28–35.