A Framework and Tool for Collaborative Extraction of Reliable Information

Graham Neubig\textsuperscript{1}, Shinsuke Mori\textsuperscript{2}, Masahiro Mizukami\textsuperscript{1}

\textsuperscript{1}Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama-cho, Ikoma-shi, Nara, Japan
\textsuperscript{2}Academic Center for Computing and Media Studies, Kyoto University
Yoshida Honmachi, Sakyo-ku, Kyoto, Japan

Abstract
This research proposes a framework for efficient information extraction and filtering in situations where 1) extreme reliability is important, 2) the amount of information to be combed through is massive, and 3) we can expect a relatively large number of human workers to be available. In particular, we are motivated by needs in times of crisis, and assume that in order to ensure the high level of reliability required, it will be necessary to have at least one human worker confirm all extracted information. Given this setting, we propose a method to improve the efficiency of manual verification by deciding which information to present to workers using machine learning techniques. Even given this efficient search framework, the amount of information on the internet is still too much for one user to handle, so we additionally create a web-based framework that allows for collaborative work, and an algorithm that allows for this framework to work on large data in real-time. We perform an evaluation using data from Twitter after the Great East Japan Earthquake, and compare efficiency using both traditional keyword search and the proposed learning-based method.

1 Introduction
In times of crisis, internet sites, and particularly social networks such as Twitter,\textsuperscript{1} overflow with information, with some reports noting an increase of activity by as much as 20 fold (Miyabe et al., 2012a). This information spans all genres, from questions or comments about the state of affairs, statements of opinion, emotional pleas, or even the spread of false rumors (Qu et al., 2009; Mendoza et al., 2010). Perhaps the information of the most interest is that which helps either crisis-responders or evacuees get a better grasp of situation (Vieweg et al., 2010), and this is particularly true when the information is provided directly by people in the disaster-affected areas (Starbird et al., 2012).

However, distinguishing useful information (e.g. “there is water at the evacuation center in Sendai high school”) from unreliable or non-actionable information (e.g. “just arrived at the evacuation center, so tired...”) takes a large amount of human effort. Luckily, however, the effort of good-willed internet users is one thing that is often plentiful in times of crisis. There have been many success stories where volunteers have banded together to turn natural language data into machine-readable format (Starbird and Stemberger, 2010), translate crisis-related information (Munro, 2010), gather survivor lists from evacuation sites and enter them into a central database (Google Japan, 2011), or even annotate data for the creation of specialized information extraction systems (Neubig et al., 2011). Given the large amount of work required in these collaborative efforts, it is common for as many as hundreds of volunteers to be involved in any single task.

On the other hand, examinations of the types of information provided on social networks after crises have shown that the number of possible information extraction tasks is large (20-30 by Corvey et al. (2012)’s classification). Information requirements also vary greatly from situation to situation, with the direction of the wind being important during the Oklahoma wildfires, and radiation measurements being important after the nuclear meltdown following the Great East Japan Earthquake (Vieweg et al., 2010; Doan et al., 2012). While a large number of volunteers may be mobilized for a single task, scaling this approach to tens or hundreds of disparate tasks has not proven

\textsuperscript{1}http://twitter.com/ retrieved on 2013-4-9.
possible given in a timely fashion. However, by increasing the efficiency of each volunteer, it is possible to reduce the overall number of volunteers needed, thus increasing the potential to tackle a much larger number of tasks in the short timeframe allowed after a disaster.

As a result, there have been a number of works that attempt to remove the requirement for manual labor by automating the information extraction process. For example, it has been noted that it may be possible to automatically identify information that contributes to situational awareness in general (Verma et al., 2011), or for more pinpoint tasks such as identifying information about safety of evacuees (Neubig et al., 2011), evacuation routes (Ishino et al., 2012), or information providers in disaster affected areas (Starbird et al., 2012). While these systems are quite promising, taking human workers out of the loop completely raises questions regarding the reliability of the information provided.

Given this background, in this work we examine a framework that enables teams of volunteers to identify useful information in a fashion that is efficient, collaborative, and highly reliable. In particular, to ensure reliability, we assume that all information provided must be checked by at least one volunteer. However, we increase the efficiency of this manual verification by learning a classifier to decide which pieces of information are likely to be relevant and should be presented to volunteers. Each time a new piece of information is labeled as either relevant or irrelevant to the task at hand, the classifier is updated to be more accurate at the task. Finally, to take advantage of collaborative work, we implement the proposed framework in a web interface that can be used collaboratively by many workers simultaneously.

Overall, while each of the individual components described below are not novel (quite standard, in fact), our work makes four major contributions: 1) combining these techniques into one over-spanning framework for efficient information extraction (Section 2), 2) proposal and evaluation of the information extraction framework in a collaborative setting, something that has not covered extensively in previous work (Section 3), 3) a relatively extensive manual evaluation of the framework on real large-scale data in times of crisis (Section 4), and 4) an open-source implementation of the proposed framework.

2 Information Filtering/Extraction Framework

As mentioned in the previous section, the overall goal of this paper is to efficiently extract reliable information from the internet. To formalize this notion, we define the target of the information extraction system as a collection of documents \( \mathcal{D} = \{D_1, \ldots, D_T\} \). From a given document \( D_i \) we would like to gather all pieces of information \( \mathcal{T}_i = \{t_{i,1}, \ldots, t_{i,J}\} \) relevant to our given task. Each piece of information is vector containing \( K \) slots to be filled \( t_{i,j} = \{t_{i,j,1}, \ldots, t_{i,j,K}\} \). In addition, we define \( \mathcal{U} = \{u_1, \ldots, u_I\} \) where each \( u_i \) corresponds to document \( D_i \) and indicates whether there is at least one piece of useful information in the document \( u_i := (|\mathcal{T}_i| > 0) \).

To give a concrete example, let us assume that the information we are interested in is evacuation areas after a crisis, and the target that we are extracting from is Twitter posts. In this case, each document \( D_i \) would be a Twitter post, and \( u_i \) would be a binary variable indicating whether there is any useful information in the post. \( t_{i,j} \) would be a set of entries about a particular evacuation site, where each column may indicate traits of the evacuation site such as “city,” “address,” and “current status.”

In the following two sections, we describe the proposed framework for finding this information in two steps: filtering, where we estimate the usefulness \( u_i \) of each document, and extraction, where we extract the information \( \mathcal{T}_i \) from documents that pass the filtering process. In particular we focus on filtering useful documents from a large document collection, and use a simple manual extraction process.

2.1 Information Filtering

We first describe two approaches to information filtering: a baseline of keyword search, and our improved method based on relevance feedback.

2.1.1 Keyword Search

Almost any attempt to find information in a large document collection will start with keyword search, where documents are retrieved according to a user query \( Q \). In the terminology that we introduced above, this means that \( u_i \) will be true ei-
ther when all of the keywords match

\[ u_i := (|D_i \cap Q| = |Q|) \]  

(1)

or when at least one of the keywords matches

\[ u_i := (|D_i \cap Q| > 0). \]  

(2)

While this technique is extremely simple, it has also proven useful in actual rescue efforts that monitor social networks for information in times of crisis (Aida et al., 2012).

2.1.2 Information Filtering using Classifiers

However, keyword search is clearly not sophisticated enough to adequately make the decision whether a particular document contains information useful to a particular task, with the “and” search in Equation (1) filtering out too many documents, and the “or” search in Equation (2) picking up too much noise. As a solution to this problem, it is common to use machine learning to create more sophisticated classifiers (Sebastiani, 2002).

Here, we overview classifiers in the case of binary classification between \( u_i = 1 \) (true) and \( u_i = 0 \) (false). In this case, we define \( N \) feature functions \( \phi_n(D_i) \) that express various characteristics of the document \( D_i \). Each feature function is assigned a weight \( \lambda_n \) and the weighted sum of feature functions is the document’s score \( s(D_i) \)

\[ s(D_i) = \sum_{n=1}^{N} \lambda_n \phi_n(D_i). \]  

(3)

In the case of \( s(D_i) \geq 0 \), \( D_i \) is classified as a positive example, and in the case of \( s(D_i) < 0 \), \( D_i \) is classified as a negative example.

In order to learn the weights \( \lambda = (\lambda_1, \lambda_2, \ldots, \lambda_K) \), a corpus of documents \( D \) is annotated with labels \( \mathcal{U}^* \), and a classifier such as support vector machines (SVMs) or naive Bayes classifiers is used to train the weight values (Joachims, 1998). In this research, we use naive Bayes classifiers as they are extremely fast to learn and perform reasonably well on document classification tasks (Dumais et al., 1998). In naive Bayes classifiers, we calculate the conditional probability of label \( u \) given the feature \( \phi_n \)

\[ P(u = 1|\phi_n) = c(\phi_n, u^* = 1) / c(\phi_n) \]

where we slightly abuse notation for clarity by defining \( c(\phi_n) \) to be the sum of all values of \( \phi_n(D_i) \) for labeled documents and \( c(\phi_n, u^* = 1) \) to be the same sum for documents labeled with \( u^* = 1 \). The probability of a label given the document is the product of the feature probabilities

\[ P(u_i = 1|D_i) = \prod_{n} P(u_i = 1|\phi_n(D_i)) / Z \]

where \( Z \) normalizes the probabilities to add to 1

\[ Z = \sum_{u_i = 1} P(u_i = 1|D_i) + P(u_i = 0|D_i) \]

We can define score \( s(D_i) \) as the log odds of \( P(u_i = 1|D_i) \)

\[ s(D_i) = \log P(u_i = 1|D_i) - \log P(u_i = 0|D_i) \]

which allows us to define each weight \( \lambda_n \) as

\[ \lambda_n = \log(c(\phi_n, u^* = 1) + \alpha) - \log(c(\phi_n, u^* = 0) + \alpha). \]  

(4)

It should also be noted that while standard classifiers are trained using the document labels \( \mathcal{U}^* \), it is also possible to directly label the features \( \phi_n \) (Melville et al., 2009; Settles, 2011). In this case, let \( l(\phi_n, u^* = 1) \) be a function that is 1 if feature \( \phi_n \) is labeled positive, and 0 otherwise. We further augment Equation (4) with a pseudo-count \( \beta \) in the case of labeled features

\[ \lambda_n = \log(c(\phi_n, u^* = 1) + \alpha + \beta l(\phi_n, u^* = 1)) - \log(c(\phi_n, u^* = 0) + \alpha + \beta l(\phi_n, u^* = 0)). \]  

(5)

In other words, if a positively labeled feature exists in a particular document \( D_i \), that document will have a higher chance of being labeled positive. This is useful in our situation, as any classifier we build will likely be combined with keyword search as described in the previous section, and the features corresponding to these keywords can automatically be labeled as positive.

The bottleneck in the construction of these classifiers is the manual creation of the labels \( \mathcal{U}^* \),
which takes a significant amount of time and effort. Fortunately, there has been some movement for disaster preparation to create labeled corpora in the crisis-related domain (Verma et al., 2011; Neubig et al., 2011; Corvey et al., 2012). However, as all pre-constructed resources, these will be necessarily limited to tasks foreseen before an actual disaster occurs, and also limited by the language of the resources (i.e. English or Japanese).

2.1.3 On-the-fly Information Filtering with Relevance Feedback

In the framework in this paper, we propose using a different approach that requires no prior creation of labels \( \mathcal{U}^* \) (and thus no foresight into the information that may be necessary in any particular situation), but also can take advantage of machine learning techniques to improve the accuracy of information filtering. In order to do so, we start with no labels and simple keyword search, but utilize the framework of relevance feedback (Zhou and Huang, 2003) to iteratively improve the classifier. The iterative process is as follows:

**Search:** From all unlabeled documents in \( \mathcal{D} \) with at least one matching keyword, the system selects \( M \) documents with the highest score \( s(D_i) \) and displays them to the user.

**Extraction/Feedback:** The user extracts information \( T_i \) from each document \( D_i \) (as described in the following section), and notes through the interface whether the document had any useful information \( (u_i^* = 1) \) or did not \( (u_i^* = 0) \).

**Learning:** Once the user finishes extracting information from the \( M \) documents, the new labels are submitted to the learning algorithm, weights are updated, and we return to step 1.

This process is extremely simple, but also satisfies a number of desiderata for our system. First, before any examples are labeled, it is possible to use simple keyword search as a starting point, so users can start search immediately without any prior labeling of data.\(^3\) However, we still have the potential to greatly improve efficiency by learning a classifier that can reduce the number of false examples \( (u_i^* = 0) \) that the user has to view. Second, the labeling criterion is extremely intuitive: if useful information that can be added to the database is found, the label is positive, and if useful information is not found the label is negative. Thus, users can essentially perform search exactly as they would normally, but the accuracy of the search results improves after each set of \( M \) documents is viewed and labeled.

2.2 Information Extraction

The final part of the framework is the process of extracting information \( T_i \) from each document \( D_i \). This is an interesting problem in its own right, with large amounts of work on automated methods (Sarawagi, 2008). There are also some works on the extraction of highly reliable information by including a human in the extraction practice by either writing regular expressions (Caruana et al., 2000), or by correcting mistakes made by automatic extraction methods (Kristjansson et al., 2004; Culotta and McCallum, 2005). While these works are relevant to our current task, our evaluation experiments are run on very short documents, for which the relevant information can manually be read, copied, and pasted into a spreadsheet with relatively high speed. Thus, in this work, the extraction of information from relevant documents is performed entirely by hand, and we leave expansion to more sophisticated methods to future work.

3 Collaborative Interface

While the method described in the previous section allows for efficient and reliable information filtering for a single worker over small-scale data, it cannot be trivially applied to larger data. The reason for this is two-fold. First, to implement the method described in the previous section, every example must be re-scored every time the classifier weights change, which results in a wait time between each example that linearly increases with the size of data at hand. Second, the previously described method assumes that there is only a single worker, but there are physical limits on the amount of data that a single worker can handle.

In this section, we further improve the framework presented in the previous section by implementing it as a streaming and collaborative information aggregation interface. The framework handles information streams by not re-scoring every possibly example, but performing greedy re-
scoring of only candidates to be presented to workers and keeping a limited number of top-scoring candidates in a memory cache. The framework is collaborative as it is implemented through a multi-user web interface that communicates with the scoring and aggregation server running in the background. An overview of the framework is shown in Figure 1.

We also show an example of the annotation interface in Figure 2. On the task page, there is a place to view and enter the keywords for the current task, and several (in this case, five) examples to be viewed and labeled by the worker. Once the worker has viewed all instances, and clicked either “+” for positive or “-” for negative, the labels can be submitted to the server to update the weights.

3.1 Work Flow from the User Side

We assume a use case where we have a small group of users, and one of the users is designated as the leader of the group.

Before the users commence the information filtering process, the group decides the type of information they want to extract (e.g. “information about evacuation areas within Miyagi prefecture”). Next, the users choose one or more keywords related to this information (e.g. “miyagi”, “evacuation”). The group leader then creates a location where all of the users can aggregate information to the specified topic using an online document management service such as Google Docs,\(^4\) or a special-purpose web site. Every user in the group then accesses the annotation interface, labels instances, and inputs the useful information found in positive instances into the location where the information is being aggregated. To ensure that multiple users are not viewing the same information, each document is only displayed to a single user.

3.2 Work Flow from the Server Side

When the group leader specifies the first keywords to be used in the task, the computation server starts to acquire examples from a web information source such as Twitter. There is a certain amount of overhead required to run even a simple classifier such as that described in Section 2.1.2, so to increase the efficiency of information retrieval we first perform a simple keyword filtering step that removes all documents that do not contain at least one instance of one of the specified keywords.

4 http://docs.google.com retrieved on 2013-4-9.
the scores in the cache will be calculated by an older version of the model. In order to solve this problem in an efficient manner, we note that even though the scores may change somewhat, in most cases the order of the scores will remain more-or-less the same as it was before the update. We further take advantage of this by greedily searching for the highest scoring example according to the following process:

1. According to the (potentially old) scores in the cache, find the highest scoring example $d_1$ and second-highest scoring example $d_2$.

2. Re-calculate $d_1$’s score according to the current version of the model.

3. Compare the score of $d_1$ to the score of the example $d_2$ according to the old model.

4. If $d_1$’s new score is higher than $d_2$’s old score, return $d_1$ as the highest-scoring example. Otherwise, re-insert $d_1$ into the cache and return to step 1.

It should be noted that this search is greedy, and thus may make mistakes when $d_1$’s score is larger than $d_2$, but not larger than examples that occur farther down in the cache.

4 Experimental Evaluation

To evaluate the effectiveness of our proposed framework and tool for extraction of highly reliable information, we perform a series of experimental evaluations. As extraction of highly reliable information is particularly important in times of crisis, we used data provided as part of the Great East Japan Earthquake Big Data Workshop including all actual Japanese tweets from Twitter for one week after the earthquake starting at March 11th, 2011, 14:45, a total of 179 million tweets.

We specify three information extraction tasks as shown in Table 1, and use these as the targets for our experiments. The first task consists of finding information about evacuation areas or rescue supplies that may be useful to those in disaster-affected areas, and the other two tasks are related to finding posts either requesting or providing information about the safety of evacuees.

Given our goal of efficient and reliable identification and extraction of information as stated in Section 1, we use as our evaluation measure the number of tweets able to be verified by workers in 30 minutes. In addition, to more closely simulate the actual situation of the tool being used in a crisis-response setting, all workers were asked to fill in a web form indicating information such as “location,” “situation,” or “name” in accordance to the information that would likely be useful for each of the tasks.

Given this data and these tasks, we perform two rounds of experiments to compare the efficiency of the learning-based interface compared to simple keyword search (Section 4.1) and the efficacy of collaborative work (Section 4.2).
4.1 Evaluation of Learning-based Information Filtering

First, we perform an evaluation of the information filtering interface described in Section 2.1.3. As features, we use character 1-to-5-grams, and a naive Bayes classifier, as it is extremely efficient to both classify and update. In Equation (5), we set $\alpha = 1$ and $\beta = 5$. As a baseline system, we use simple keyword search. All results were provided by a single user who had time to practice using the interface before results were recorded.

We show the results for the three tasks in Figure 3. From the results indicating the number of pieces of useful information extracted on the left side of the graph, we can see that the proposed learning capability improves the efficiency, with increases ranging from 35%-159%. This increase can be largely attributed to an increase in the information filtering accuracy, or the number of documents displayed to the user that have at least one piece of useful information. The rolling average of accuracy is shown on the right side of Figure 3.

We can see that there is a significant difference in the information filtering accuracy between tasks, and this affects the gain afforded by the learning capability. Specifically, “Safety Info. Provision” has lower accuracy than the other tasks, largely because it is difficult to distinguish between provision of information (“I heard that XXX is safe”) and requests for information (“I wonder if XXX is safe”), while the latter is much more common in the corpus (approximately five times according to Murakami and Hagiwara (2012)’s estimate). Thus, for more difficult tasks improving the accuracy of the classifiers could lead to further improvements in the gains provided by the proposed technique.

4.2 Evaluation of the Collaborative Interface

In addition, to evaluate the collaborative web interface described in Section 3, we performed experiments in which multiple annotators worked collaboratively on an information filtering task. The experimental setup is identical to that described in the previous section, but we focus only on the Evacuation/Rescue Supplies task. As a comparison, we compare results for when 1, 2, or 3 users work collaboratively on a single information filtering task, performing two experiments for each number of users.

The result of this experiment is shown in Figure 4. After 30 minutes of work, a single user had extracted an average of 43 pieces, two users had extracted 103 pieces, and three users had extracted 129 pieces of useful information and added them to the shared aggregation site. Thus, we can see that increasing the number of users results in an approximately linear increase in the amount of information extracted, confirming the effectiveness of allowing multiple users to work on a single task, and share the results of labeling with a single classifier. As each worker works largely independently, we hypothesize that this trend will continue for even larger numbers of users.

In Figure 5 we show the improvement in efficiency of information extraction as each run progresses. From the graph, we can see that in all cases, the efficiency at the end of the run has increased by 1.3-2.0 times over that achieved in the initial five minutes. Figure 6 displays the rolling average of positive examples, and we can see that

---

Character $n$-grams remove the effect of analysis mistakes that occur when performing Japanese tokenization on non-standard text such as Twitter.
the percentage of positive examples labeled increases drastically over time, with accuracy near 100% achieved in four out of six trials, and all trials achieving accuracy over 60%. However, compared to this large increase in the accuracy of the examples presented to users, the increase in the amount of information extracted is small. This is because even after positive information has been identified, there is a small but fixed amount of work required to enter the useful information into the information aggregation site. As a result, we can expect that further improvements in information extraction efficiency can be achieved by automatically extracting candidates to fill in each column of information to be extracted, and make it possible for a human to simply press a button to verify the information if it happens to be correct.

5 Conclusion/Future Work

In this paper, we presented a framework to efficiently, reliably, and collaboratively filter and extract useful information from the large and noisy web, with a focus of information extraction from Twitter during crisis situations. As a result, we found that the proposed framework led to an increase in efficiency of 35%-159% over simple keyword search, with further gains possible when more than one user participates in the information extraction process.

As future work, we can think of expansions to multi-class information extraction problems. In this paper, we limited our experiments to situations where each classifier is trained to identify a single type of information, so identifying three types of information will require three separate rounds of classifier training. While this is a simple setup for users to understand, it is inefficient in the case of a large number of classes, so it is worth examining the possibilities of extracting multiple types of information in a single process.

Another interesting line of work is the provision of extracted information in an easy-to-consume form for people in disaster areas through a QA system that can be accessed through telephone when other communication tools such as the internet are not available (Kazama et al., 2012).

Assessing the reliability of information on the web is an important challenge, particularly in times of crisis (Mendoza et al., 2010). Work to automatically assess the reliability of information may focus on classifiers assessing the text, user, topic, and dispersal patterns of the said information (Castillo et al., 2011) or comments on social networks casting doubt on said information’s veracity (Miyabe et al., 2012b). These methods could be combined with our information extraction method to further ensure the reliability of the extracted information.

There are also a number of improvements that could be made to the extraction algorithm itself. For example, while in this work we used a simple Naive Bayes classifier, there are classifiers developed specifically for the task of classifying positive examples (Schölkopf et al., 2001), which may increase the accuracy of information identification. Other promising directions include the application of more advanced information extraction techniques, the identification of information that is identical to information that has already been extracted, or application to crowd-sourcing platforms such as Mechanical Turk.
Acknowledgments

The authors sincerely thank Shingo Murakami for his assistance in developing the web interface during the Great East Japan Earthquake Big Data Workshop. We would also like to thank all of the subjects who lent their time to participate in manual annotation experiments, and the organizers of the Great East Japan Earthquake Big Data Workshop and Twitter for providing the data that made the experiments possible.

References

Shin Aida, Yasutaka Shindoh, and Masao Utiyama. 2012. Regarding the creation of the “Great East Japan Earthquake rescue request information extraction site” and rescue activities (in Japanese). In Proc. 18th NLP.


Kate Starbird and Jeannie Stamberger. 2010. Tweak the tweet: Leveraging microblogging proliferation with a prescriptive syntax to support citizen reporting. In Proc. ISCRAM.


