Improving Classification of Crisis-Related Social Media Content via Text Augmentation and Image Analysis

SHIVAM SHARMA, New Jersey Institute of Technology
CODY BUNTAIN, New Jersey Institute of Technology

Additional Key Words and Phrases: Incident Streams, TREC, TRECIS, crisis informatics

1 INTRODUCTION

Identifying, classifying, and prioritizing crisis-related content in social media is an increasingly important task, as users of online platforms continue to expect emergency-response officials to monitor these channels. Much of the current work in this area, however, focuses on applications of neural language models (NLMs) to the text of these social media messages, leaving many meta-content and multi-modal signals unaddressed. This work challenges this text- and NLM-centric view as it applies to crisis informatics and the Incident Streams track at the annual Text Retrieval Conference (TREC-IS) by first measuring the performance enhancements NLMs provide over classical text-classification pipelines and then by integrating external data sources and non-textual image analysis. Results suggest classical machine learning (ML) models are still competitive to NLMs, especially in identifying high-priority content, but adding a simple text augmentation strategy results in significant gains in NLM performance. Preliminary results are consistent with the community's focus on NLMs as our work suggests augmented NLMs perform best in classification, while integrating image analysis has marginal effects on performance. Augmenting samples with inferences from CrisisMMD, however, also significantly increases prioritization performance, suggesting strategies for integrating other data and signals remain valuable.

To justify these claims, this paper presents a systematic comparison of classical ML and NLM models for classifying and prioritizing social media content. This paper spans TREC-IS 2020-A and 2020-B, and as such, we are able to validate these various approaches against a held-out set of numerous crisis events and against a collection of COVID-19 events. In this context, we compare our classical ML system from prior TREC-IS editions to several NLM-based systems. We also present a system that integrates image analysis into automated prioritization, for which we currently lack official TREC-IS performance metrics.

In particular, this paper describe our research around TREC-IS and answer the following research questions:

(1) Which neural language model performs best for TREC-IS tasks?
(2) Does a simple synonym-substitution strategy for augmenting textual training data improve classification performance?
(3) Does integrating image analysis improve classification performance?
(4) Does augmenting image data with crisis-related media improve classification performance?

Primary contributions of this work will be of interest to those studying crisis informatics and methods for integrating multi-modal content into models of online behavior. Crisis informatics researchers in particular may benefit from the analysis of imagery and simple methods for boosting training data given the overall rarity of crisis-level events.

Authors' addresses: Shivam Sharma, New Jersey Institute of Technology, ss4354@njit.edu; Cody Buntain, New Jersey Institute of Technology, cbuntain@njit.edu.

2 METHODS

This section outlines four NLM-based pipelines for classifying and prioritizing content provided from the TREC-IS track along with the a classical ML baseline from prior TREC-IS editions [1]. NLM-based systems have shown themselves superior in general natural language processing tasks, and numerous such models have become popular in recent years (e.g., BERT, XLNet, RoBERTa, etc.). Prior work on TREC-IS has suggested classical machine learning approaches with strong feature engineering perform as well as neural models, but as the TREC-IS dataset grows, this result may no longer hold. We therefore test a select of neural methods, as each presents slightly different text analysis approaches. We begin by describing our classical ML approach, which we then compare against a collection of NLM-based systems. From these models, we then implement a series of increasingly sophisticated models by integrating external and multi-modal data. First though, we describe the evaluation metrics used to compare our models.

2.1 Evaluation Metrics

To compare the different approaches described below, we leverage the standard metrics of precision, recall, and F1 for both the information-type and priority classifications, macro-averaged across each event. We additionally include the metrics produced by the TREC-IS 2020-A evaluation notebook, which includes both an nDCG metric for ranking content by priority and divides F1 into two sets, one restricted to "actionable" information types, and the other containing all possible labels. This "actionable" set is restricted to the top six information types with the highest average priority score, making them the most "important" types to classify correctly in a qualitative sense.

2.2 A Baseline Classical ML Model

In this classical ML baseline, we rely on standard text-mining techniques. To convert tweets into feature vectors, we first extract a collection of Twitter-specific features to create a 15-dimensional numeric vector, including capitalization, whether the account is verified, whether the tweet is a retweet, the tweet's sentiment (using VADER [2]), as well as number of characters, hashtags, media links, mentions, and terms. These features capture meta-data aspects of these messages.

To capture content features, we use a standard bag-of-words model. We tokenize each tweet using NLTK's tweet tokenizer and apply a term-frequency, inverse-document-frequency (TF-IDF) weighting to the top 10,000 features. To weight these features, we learn this TF-IDF feature weighting using a subsample of English tweets collected from Twitter's public sample stream between 2013 and 2016; we learn on this larger sample to increase generalizability in our data and address out-of-vocabulary issues in the original TREC-IS set. In this vectorizer, we create unigrams and bigrams and remove rare terms (those that occur fewer than four times) and common terms (those that occur in more than half of the documents).

From these features, we train a naive Bayes classifier using a one-versus-rest model to predict multiple information-type labels for each tweet. We similarly train a random-forests classifier to predict whether a tweet contains high- or low-priority messages.

2.3 Choosing an NLM Framework

Prior to implementing our NLM systems, we first had to evaluate and determine which extant NLM to use (as many pre-trained models are now available online). We shortlisted four models, namely BERT, RoBERTa, XLNet and DistilBERT, for testing on the 2019-B data and chose the best models from these as our main models for both the submissions. Each

of these systems is an NLM and uses the `simpletransformers` library, which is based on the HuggingFace's Library for transformers [4].

For evaluating the performance of the models, we used a 90-10 train-validation split on the training dataset, with 90% training data and 10% validation data. We evaluated the model on accuracy and F1-score. After evaluations on the selected models we decided, through analysis on F1-score and accuracy, to use RoBERTa for the Information Type classification task and BERT for the Priority Type classification task.

2.4 Training and Pre-processing

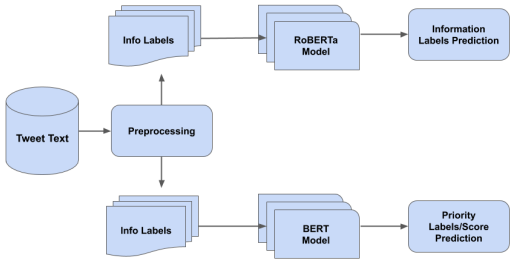
After NLM selection, these models are retrained using the training data. The pretrained weights are loaded to the model architecture and then the model is retrained for 10 iterations on the training data. The simpletransformers library takes as input a pandas dataframe with two columns, one with the text and one with labels. The predict function of the simpletransformers library model object outputs two lists, one containing the raw model outputs for every text in the testing data, and the other containing the class predictions for every text in the testing data. For 2020-A, since we were required to submit the classes for each text, the predictions were used and for 2020-B, in which we were required to submit the weights for each class for every text, the raw outputs passed through a softmax function were used. All the pipelines initially begin with preprocessing of the tweet text in the training data, which mainly includes data cleaning and refining. The text is first cleaned of all punctuations and special characters like #, @, etc. We replace all the URLs, mentions and hashtags from the text with "URLHERE", "MENTIONHERE" and "HASHTAGHERE" tags, respectively.

2.5 Four TREC-IS Participant Systems

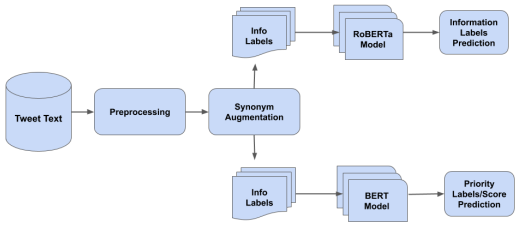
Our classical ML system. which we describe below, provides a strong baseline for this task, as a version of this system was used in prior TREC-IS editions [1]. We then describe four systems we have submitted to TREC-IS 2020-A and 2020-B, which we summarize in Figure 1.

2.5.1 A Simple Synonym-Augmentation Strategy. After analyzing the data we found a heavy imbalance between labels/categories which are considered as High Importance vs those which are considered as Low Importance, with a majority belonging to the Low Importance information category. To counter this, we considered using the oversampling strategy SMOTE to increase the number of Critical/High Importance Information Type categories. After a few tests of comparing different scores like precision, recall and F1 score for different oversampling strategies like SMOTE through imbalance-learn library, we decided on "generating" new tweets of High Importance Information categories. The text augmentation was done on the idea of replacing specific words in a tweet with their synonyms, so as to "generate" new tweets for training data which do not deviate from their original meaning but still have some semantic differences from the original tweet. Since the tweets are related to disasters and calamities, we decided to replace the verbs with their synonyms as those would be the words which would have higher weight in the sentence. After some test, we decided on replacing one single verb with four synonyms, as more than that and the synonym would sometimes change the meaning of the text. This approach gave us more than four thousand new tweet texts to be used in training with Critical/High Importance Information type labels. This approach can also be seen as a form of weak-supervision, through which new samples are generated and labels applied without direct supervision of results; other platforms like Snorkel perform similar augmentation strategies.

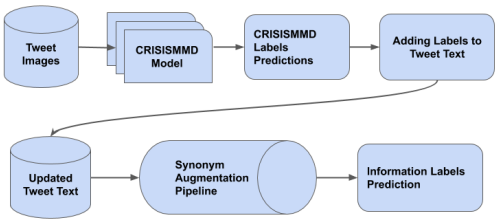
2.5.2 CrisisMMD-based model. Introduction of more information for every datapoint improves the performance of a model. Introducing significant keywords in a textual dataset which correlate to the classes improves the performance of



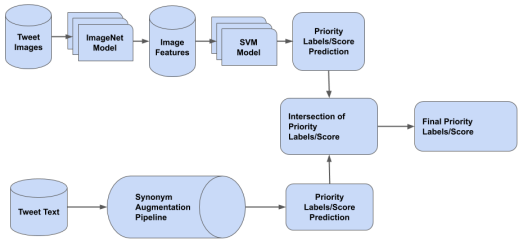
(a) Core NLM Model.



(b) NLM Model with Simple Synonym Augmentation Strategy.



(c) Augmentation with CrisisMMD Text Labels.



(d) Image Analysis for Priority.

Fig. 1. Four NLM-, Augmentation-, and Multi-Modal-Based TREC-IS Participant Systems.

a textual classification model. This was the basic idea behind the CrisisMMD pipeline. To introduce keywords in the tweet text which in theory would correlate with the information type classes. Since many tweets include media shares like images, we thought of using these images and getting information from them. CrisisMMD [3] is a dataset of tweets during disasters with the main focus towards the images in such tweets. One of the tasks is to classify a given tweet into 8 humanitarian classes, namely, Affected Individuals, Infrastructure and utility damage, Injured or dead people, Missing or found people, Rescue, volunteering or donation effort, Vehicle damage, Other relevant information and Not humanitarian. These classes are very similar to TREC-IS Information Type classes.

We used the VGG16 model architecture provided by Ofli et.al. [3] retrained on the CRISISMMD training dataset. The model architecture comprises the last layer of the VGG16 model with 'imagenet' weights, which was passed through a dense layer for predicting the humanitarian labels. An Adam optimizer with a learning rate of 1e-6 was used, and a categorical crossentropy was used as the loss function. This model was then used to predict humanitarian labels for all the images present in the training dataset. These labels were then added to the text of the tweet the image was taken from. This new dataset was then passed through the synonym-augmentation pipeline. The introduction of these labels to the data showed comparable results in overall and low importance information type labels but a noticeable improvement in the high importance type labels.

2.5.3 Augmenting with Image Data. A non-trivial portion of social media data in the TREC-IS dataset includes media shares, and high-priority information often includes image data (though not all media shares are high-priority). Hence, augmenting our models information type and priority may improve classification performance and better reflect a human annotator's assessment of the data. Here, we describe how we select image data from the TREC-IS dataset, featurize it, and augment our extant text-based models with this data. We anticipate the image data presents a different view of the data, so we integrate our image- and text-based pipelines by taking the union of all labels generated by both pipelines.

Our initial results from the synonym-augmentation pipeline showed promising results. To improve on this pipeline further we thought of using the media in the tweets, specifically the images. We created an image pipeline to predict labels for the tweets with images and then added those labels to the labels predicted by the synonym-augmentation pipeline. For the information type labels we did a union of the labels predicted by the synonym-augmentation pipeline. For the priority label/scores we did an intersection between the labels/scores, that means, we chose the one with higher priority between the two. We get the images from all the tweets containing images and get their features using the VGG16 pretrained model. These features are then split into training and test/validation sets. A Linear SVC model with default parameters is trained on the training data and predictions are made on the test/validation set. For 2020 B, we needed to submit scores instead of labels for the Priority task, so in place of the Linear SVC model, we used a Linear SVR model. An initial preprocessing for all the images includes resizing the images to (224,224). The model used has the same architecture as the VGG16 model used in CRISISMMD pipeline, with just the prediction layer removed. This means that the features generated are the last layer of the VGG16 model with 'imagenet' weights. The labels predicted by the Image Pipeline and the labels predicted by the Synonym-Augmentation Pipeline were compared and the higher priority of the two was selected as the final label. For 2020 B submission scores, a similar approach was taken, where the scores from both the pipelines were compared and the higher one of the two was taken as the final score. To evaluate the results of this pipeline a 10 fold cross-validation was performed with accuracy and F1-score as evaluation scores. There was marginal improvement in the overall F1 scores, but the cross validation showed promising results in the priority task, for which this pipeline was used.

3 RESULTS

3.1 Comparing Classical ML and NLM Models

Table 1 present the performance metrics of a collection of NLM models *along with* the baseline ML model, as applied to the 2020-A test data released at the end of 2020-A. Readily apparent from these results is that all NLM models significantly outperform the classical ML model.

Table 1. Baseline ML and NLM Model Performance Analysis, on 2020-A Test Data. Bolded cells are the best-scoring systems.

Model	nDCG	All-Type F1	Actionable F1	Priority F1
ML Model	0.413	0.153	0.099	0.182
BERT	0.450	0.119	0.095	**0.226**
RoBERTa	**0.475**	0.158	0.079	0.192
XLNET	0.370	**0.197**	**0.120**	0.148
DistilBERT	0.451	0.123	0.095	0.210

3.2 Evaluating Synonym Augmentation

Next, we compare our synonym-based augmentation strategy for the NLM models shown above, as shown in Table 3. This table shows two main results: First, our selection of RoBERTa as the base pre-trained model appears suboptimal, as BERT and XLNet appear to outperform RoBERTa by small margins. Second, comparing these multiple models trained on synonym augmentation to results from Table 1 above, synonym augmentation boosts performance in each metric except nDCG. These results suggest augmentation is a useful strategy for dealing with class imbalance, and NLM models improve in accuracy through its application.

Table 2. Synonym Augmentation Performance, 2020-A Test Data. Bolded cells are the best-scoring systems.

Model	nDCG	All-Type F1	Actionable F1	Priority F1
BERT with Synonym-Augmentation	**0.474**	0.130	0.127	**0.228**
RoBERTa with Synonym- Augmentation	0.462	0.208	0.146	0.177
XLNET with Synonym- Augmentation	0.451	**0.228**	**0.161**	0.161
DistilBERT with Synonym- Augmentation	0.443	0.129	0.128	0.201

As part of the 2020-A evaluation, we submitted both the RoBERTa+BERT text system and systems trained using this augmentation strategy. Table 4 show these results across the three 2020-A tasks (Task 1 is the full info-type classification set, Task 2 is evaluated only on a restricted set of info-types, and Task 3 applies models to unseen COVID-19 events). For comparison, we also include the alternate top model as reported by the TREC-IS organizers for 2020-A. Synonym augmentation enhanced performance over the core NLM models in nDCG and information-type F1 in Task 1 and Task 2, but we see a penalty in prioritization with augmentation. Surprisingly, augmentation also seems to have decreased performance in the COVID-19 task in both F1 metrics while nDCG improves in the COVID-19 context. These results suggest our augmentation scheme should not be used for estimating prioritization.

Table 3. Results from TREC-IS 2020-A.

Model	nDCG	Actionable F1	Priority F1
Task 1			
Best TREC-IS Metric	0.4866	0.1674	0.2630
Base NLM Model	0.4632	0.0792	0.1524
NLM + Augmentation	0.4776	0.1466	0.0958
Task 2			
Best TREC-IS Metric	0.3960	0.1504	0.1939
Base NLM Model	0.4609	0.1131	0.1554
NLM + Augmentation	0.4784	0.1151	0.1004
Task 3			
Best TREC-IS Metric	0.2662	0.1540	0.2309
Base NLM Model	0.2289	0.1106	0.2765
NLM + Augmentation	0.2352	0.0993	0.2177

3.3 Comparisons Across NLM-Model Strategies

Finally, we now turn to an evaluation of the three models we submitted to 2020-B. We first report the performance metrics using the 2020-A test data, as shown in Table 5.

Table 4. Synonym Augmentation Performance, 2020-A Validation Data. Bolded cells are the best-scoring systems.

Models	nDCG	All-Type F1	Actionable F1	Priority F1
Synonym-Augmentation	**0.478**	**0.209**	0.147	**0.208**
CrisisMMD Integration	0.407	0.187	**0.161**	0.184
Image Analysis	0.457	0.19	0.125	0.207

Overall we can see minimal improvement by the Image Pipeline and the CrisisMMD Pipeline, but the evaluations of the actionable class shows that the CrisisMMD pipeline is outperforming our best model from 2020-A evaluation. The Image Pipeline also outperforms our synonym-augmentation pipeline in the priority scores. These results motivated us to use, for 2020-B submissions, the CrisisMMD pipeline for the Information Type Classification and Image Pipeline for Priority Score predictions.

Tables 6 and 7 show results returned by the TREC-IS organizers. For 2020-B Task 1, the best-performing system outperforms all four of our submissions, whereas for Tasks 2 and 3, the NLM-based synonym-augmentation approach is consistently a top-performing system. Incorporating image analysis, however, seems to provide little in the way of performance gains over other approaches, exceeding other models only in the nDCG metric in Task 2 (and by a small enough margin as to be attributable to measurement error).

Table 5. Results from TREC-IS 2020-B, Tasks 1 and 3.

Model	nDCG	Actionable F1	Priority F1
	Task 1		
Best TREC-IS Metric	**0.5032**	**0.3215**	**0.2582**
Synonym-Augmentation	0.4479	0.2634	0.2029
CrisisMMD Integration	0.4218	0.1879	0.1417
Image Pipeline	0.4473	0.1879	0.2029
Classical ML Pipeline	0.4170	0.1712	0.1064
	Task 3		
Best TREC-IS Metric	**0.4559**	**0.1629**	**0.2551**
Synonym-Augmentation Pipeline	0.4343	**0.1629**	**0.2551**
CrisisMMD Integration	0.3969	0.1590	0.2544
Image Pipeline	0.4325	0.1590	0.2551
Classical ML Pipeline	0.4267	0.0210	0.1375

Table 6. Results from TREC-IS 2020-B, Task 2.

Model	nDCG	Info-Type F1	Priority F1
Best TREC-IS Metric	**0.4479**	**0.2548**	**0.1838**
Synonym-Augmentation	0.4468	**0.2548**	**0.1838**
CrisisMMD Integration	0.4206	**0.2548**	0.1708
Image Pipeline	**0.4479**	**0.2548**	**0.1838**
Classical ML Pipeline	0.4152	0.1713	0.1162

4 FUTURE WORK AND CONCLUSIONS

Our experiments confirmed that the neural models work better than our original ML baselines. Our method of using synonyms for augmentation showed improvements in the results. The label inclusion from CrisisMMD dataset and the label-union method of the image pipeline also marginally improved the performance. Our aim for developing these pipelines is to test different models for better performance. Since our experimentation with additional data inclusion seems to improve results, we plan on adding additional data like links in the media shares and location data.

Our work on improving the initial BERT models revolved around the idea of introducing additional data to the model, whether through synonym-augmentation or by adding CrisisMMD labels or the union method for the image pipeline, we could conclude that additional data improved our results.

Our future work, thus, also revolves around introducing additional data to the models. Our idea with the synonym-augmentation model was to counter the data imbalance. We aim to build up on that idea through the use of text generation models like GPT3 to further improve the learning of our models. GPT3 uses few-shot, one-shot and zero-shot learning models, which gives us a chance to improve the data without worrying about the amount of data required to customize a model.

Another idea we intend to work on is using location of where a tweet was made from and/or location of a place mentioned or tagged in the tweet. We hope that the location of a tweet should be helpful in determining in which informational category the tweet belongs to. For example, a tweet made from New Jersey about the wildfires in California has a higher probability of being in a lower priority information class than a tweet made from California.

The informational type classes are further divided into 4 major topics, namely, Request, CallToAction, Report and Other. These topics should help in providing some hierarchical information for the information class. We aim to use this information to further improve on our models.

Our experiments with NLMs have shown that they do outperform the classical ML models. The differences in the evaluations for validation and test data points out the need for a cross validation for all the models. Furthermore we plan on evaluating results on more models and choose the best from a wider range of models.

REFERENCES

[1] Cody Buntain. 2018. *Learning Information Types in Social Media for Crises.* TREC Notebook. NYU, Social Media and Political Participation Lab. http://dcs.gla.ac.uk/~richardm/TREC_IS/2018/Papers/trecis2018.umd_hcil.notebook_paper.cbuntain.pdf

[2] CJ J. Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. *Eighth International AAAI Conference on Weblogs and …* (2014), 216–225. http://www.aaai.org/ocs/index.php/ICWSM/ICWSM14/paper/view/8109{%}5Cnhttp://comp.social.gatech.edu/papers/icwsm14.vader.hutto.pdf

[3] Ferda Ofli, Firoj Alam, and Muhammad Imran. 2020. Analysis of Social Media Data using Multimodal Deep Learning for Disaster Response. In *17th International Conference on Information Systems for Crisis Response and Management.* ISCRAM Society, Blacksburg, VA.

[4] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv* (2019), arXiv–1910.