Keyphrase Extraction from Disaster-related Tweets

Jishnu Ray Chowdhury
Kansas State University
Manhattan, KS
jishnurayc@ksu.edu

Cornelia Caragea
University of Illinois at Chicago
Chicago, IL
cornelia@uic.edu

Doina Caragea
Kansas State University
Manhattan, KS
dcaragea@ksu.edu

ABSTRACT

While keyphrase extraction has received considerable attention in recent years, relatively few studies exist on extracting keyphrases from social media platforms such as Twitter, and even fewer for extracting disaster-related keyphrases from such sources. During a disaster, keyphrases can be extremely useful for filtering relevant tweets that can enhance situational awareness. Previously, joint training of two different layers of a stacked Recurrent Neural Network for keyword discovery and keyphrase extraction had been shown to be effective in extracting keyphrases from general Twitter data. We improve the model's performance on both general Twitter data and disaster-related Twitter data by incorporating contextual word embeddings, POS-tags, phonetics, and phonological features. Moreover, we discuss the shortcomings of the often used F1-measure for evaluating the quality of predicted keyphrases with respect to the ground truth annotations. Instead of the F1-measure, we propose the use of embedding-based metrics to better capture the correctness of the predicted keyphrases. In addition, we also present a novel extension of an embedding-based metric. The extension allows one to better control the penalty for the difference in the number of ground-truth and predicted keyphrases.

CCS CONCEPTS

• **Information systems → Information retrieval**; *Data mining*;

KEYWORDS

Keyphrase Extraction, Social Media Analytics, Disaster, Twitter

ACM Reference Format:
Jishnu Ray Chowdhury, Cornelia Caragea, and Doina Caragea. 2019. Keyphrase Extraction from Disaster-related Tweets. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3308558.3313696

1 INTRODUCTION

Keyphrase extraction is the task of automatically extracting words or phrases from a text, which concisely represent the essence of the text. Keyphrases can be used for multiple tasks including text summarization, classification, and opinion mining. They can also be highly valuable for retrieving relevant documents when searching for specific information. In particular, keyphrases can be used to organize and retrieve useful information from Twitter.

Twitter is an online social media platform, where people communicate via short, terse messages ("tweets"). Due to the ease of posting a tweet, and the popularity of the platform, Twitter can provide up-to-date news around the world, faster than standard media outlets. Thus, in recent years, Twitter has become an important source of real-time information [55, 61]. Specifically, during crisis situations, Twitter has revolutionized the way in which affected populations communicate with response organizations [6, 8]. On one hand, the affected populations can follow and spread the latest updates from government and response organizations on Twitter. On the other hand, affected individuals can post information that can be used to enhance situational awareness and, in some cases, save lives. Emergency hotlines, such as 911, could be overwhelmed by the large number of calls received during major disasters, and as a consequence, individuals often turn to social media, including Twitter, to request assistance [14, 23, 29, 45]. For example, a North Carolina family was rescued during Hurricane Florence after posting the following message on Twitter: *'If anybody could help… our car is under water and so is our house. Stuck in the attic. Phone is about to die. Please send help'* [23].

While social media can help enhance situational awareness, facilitate rescue operations, and ultimately save lives, its value is highly unexploited by response teams, in part due to the lack of tools that can help filter relevant, informative, and actionable information posted during disasters [60]. To address this limitation, the annotation of tweets with informative keyprhases can enable filtering of actionable information by the disaster management and response teams, in real time. In fact, in Twitter, hashtags assigned directly by the users posting the tweets are sometimes used to annotate tweets and can be seen as keyprases useful for filtering. However, a vast majority of tweets do not have any hashtags associated with them, as illustrated by a dataset we crawled during several crisis events that happened in the fall of 2017. Thus, to speed up the search and retrieval of potentially actionable disaster tweets posted by affected individuals, there is an urgent need for methods that can automatically extract keyphrases from tweets without primarily relying on hashtags. The extracted keyphrases can potentially be recommended to users as they are typing their messages, and thus a smaller number of relevant tweets would be retrieved by systems that facilitate filtering of information for the response teams.

Keyphrase extraction from tweets, however, is a non-trivial task due to the tweets' informal nature, noisiness and short length (currently, tweets have a limit of 280 characters). These factors limit the utility of many of the existing unsupervised keyword extraction techniques, which are designed to work on longer text using term-frequencies, word co-occurrences, and clustering methods. Recently, researchers started to explore keyphrase extraction from single tweets using deep-learning models. For example, Zhang et al.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313696

[63] proposed a model based on joint-layer Recurrent Neural Networks (RNNs) and achieved an F1-score of 80.97% when used to extract keyphrases from the general Twitter. However, this RNN-based model has not been extended to extract keyphrases from tweets posted in a specific context, e.g., during a disaster. In fact, there is a distinctive lack of works for extracting keyphrases from disaster tweets. This task is especially challenging as the vocabulary used during a disaster is focused and limited, as compared to the vocabulary in a general collection of tweets. Furthermore, the keyphrases extracted should reflect information related to the event, people involved and their locations, the type of assistance needed, the type of situational awareness provided, etc. Thus, our research agenda is designed to answer the following questions:

(1) *Can we enhance the existing model to improve its performance on both disaster tweets and general tweets?*
(2) *How do the joint-layer-Recurrent Neural Network based models trained on general tweets perform on disaster tweets?*
(3) *How well do the joint-layer-Recurrent Neural Network models perform when trained and tested on disaster tweet data?*

Regarding question (1), we discovered that adding contextual word information, phonetics and phonological features, along with part of speech (POS) tag information, tends to increase the performance of the model on both the general domain and the disaster domain, when training the model on data from the relevant domain. Regarding question (2), it is expected that the model will not perform very well if trained on a general-domain Twitter data and tested on a specific-domain data (e.g., disaster data). However, considering that the general training data and the specific disaster test data are both collected from Twitter, it is still interesting to empirically find the degree to which the RNN-based model trained on general Twitter data can or cannot generalize to disaster data. Finally, regarding question (3), while it is expected that the model will have better performance when trained and tested on specific disaster tweet data, it is interesting to understand how significant the improvement is and what additional information helps the most.

While investigating the above research questions, we found that the exact match F1-based evaluation is not ideal for determining the quality of the extracted keyphrases. As an alternative to the exact match F1 metric, we explored embedding-based similarity metrics [11, 12, 35, 48], and proposed a novel extension to symmetric greedy embedding based similarity scoring. The extension allows for more fine-grained control over the penalty for the difference in the number of ground-truth and predicted keyphrases.

We describe our extensions to the original RNN-based model for keyphrase extraction [63] in Section 3. In Section 4, we present the details of our datasets. We describe the experimental setting in Section 5.1, and the details of our embedding-based evaluation metric in Section 5.2. Finally, we present and discuss our results in Section 5.3 and conclude the paper in Section 6.

2 RELATED WORK

Keyphrase extraction can be supervised or unsupervised. Supervised approaches treat keyphrase extraction as a classification problem. In supervised approaches, a model is trained to learn to classify keyphrases from training data that is annotated with keyphrases [7, 13, 18, 28, 56, 57, 62]. Many unsupervised keyphrase extraction techniques had also been previously proposed [10, 15, 24, 27, 33, 47, 49, 53]. They usually extract candidate keyphrases and rank them based on term frequencies, word co-occurrences, and other similar features. However, the length of tweets is usually too short for most of these techniques to be applicable. Currently, tweets are limited to a length of 280 characters, with the average length of the tweets in a collection being even smaller. Furthermore, keyphrases will rarely occur more than once in a single tweet. These issues along with the noisiness of tweets can make keyphrase extraction from social media platforms such as Twitter a difficult task.

Some of the previous works on keyphrase extraction from Twitter [5, 66] take the approach of inferring keyphrases from multiple tweets. Thus, they can only extract some topical information and trends that are common to multiple tweets. At the level of multiple tweets, some of the word-count based approaches for keyphrase extraction can be still applicable. However, our target is to extract keyphrases from single tweets. Along this line, the approaches proposed by Marujo et al. [30] and Zhang et al. [63] work for single tweets. Marujo et al. [30] showed that word embeddings in a system such as MAUI [31] performs better than the TF-IDF baseline [53]. Furthermore, the Joint-Layer-RNN model proposed by Zhang et al. [63] was shown to be even better than the model proposed by Marujo et al. [30]. The joint-layer RNN model can predict keyphrases of any arbitrary length, based on single tweets, without any significant pre-processing or hand-engineered features. Thus, we chose to focus on this model in our study.

Regarding retrieval and filtering of tweets during disasters, there are many previous works related to processing social media during emergency situations [19], extracting information from Twitter to enhance situational awareness [21, 58, 59], and classifying or clustering disaster-relevant tweets [3, 54, 64], among others. However, to the best of our knowledge, no existing works have focused specifically on the extraction of keyphrases from disaster tweets, a task that we focus on in this study.

Finally, in terms of evaluation, there are previous works related to embedding-based similarity metrics [11, 12, 35, 48], which we also investigate. These metrics have been used for paraphrase detection, assessment of student input [48], and other similar tasks including assessment of the quality of generated dialogues [25, 51] by conversational models. We show that embedding-based similarity metrics can also be used to compare the quality of predicted keyphrases with respect to the ground truth keyphrases. Our specific evaluation method is based on the symmetric greedy matching embedding-based metric, which was introduced in [48].

3 APPROACH

Recurrent Neural Networks (RNN): All models investigated in this work are based on RNN, and are inspired by the model proposed by Zhang et al. [63]. Given an input sequence, $(x_1, x_2, x_3, ..., x_n)$, an RNN processes each token x_t at every time-step t by producing a hidden state, h_t, using the previous hidden state, h_{t-1}, from the previous time-step $t - 1$. Formally, this can be expressed as:

$$h_t = f(x_t, h_{t-1}) \tag{1}$$

where f is a non-linear activation function. The final output of an RNN can be a sequence of hidden states, $(h_1, h_2, h_3, ..., h_n)$. We used a specific type of RNN, called Long-Short Term Memory (LSTM) network [17]. An LSTM is equipped with three gates (an input gate, i, an output gate, o, and a forget gate, f), along with a cell-state, c, and a hidden state, h. The function of an LSTM at each time-step, t, given the input, x_t, can be described with the following equations:

$$f_t = \sigma(x_t W_f + h_{t-1} U_f + b_f) \tag{2}$$
$$i_t = \sigma(x_t W_i + h_{t-1} U_i + b_i) \tag{3}$$
$$o_t = \sigma(x_t W_o + h_{t-1} U_o + b_o) \tag{4}$$
$$g_t = \tanh(x_t W_c + h_{t-1} U_c + b_c) \tag{5}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{6}$$
$$h_t = o_t \odot \tanh(c_t) \tag{7}$$

where $x_t \in \mathbb{R}^d, h_t, c_t \in \mathbb{R}^h$, $W_f, W_i, W_o, W_c \in \mathbb{R}^{dxh}$, $U_f, U_i, U_o, U_c \in \mathbb{R}^{hxh}$, $b_f, b_i, b_o, b_c \in \mathbb{R}^h$ (h represents the number of hidden units, and d represents the dimension of the input), σ is usually the sigmoid activation function, and $\odot$ is the Hadamard product.

The LSTM that we used is bidirectional [16], in that it has two encoders: one forward and one backward. The forward encoder starts processing the input sequence $(x_1, x_2, x_3, ..., x_n)$ from left to right, thus, creating a forward hidden state representation $(\overrightarrow{h_1}, \overrightarrow{h_2}, \overrightarrow{h_3}, .., \overrightarrow{h_n})$ of the input sequence. The backward encoder starts processing the input sequence from right to left, thus, creating a backward hidden state representation $(\overleftarrow{h_1}, \overleftarrow{h_2}, \overleftarrow{h_3}, .., \overleftarrow{h_n})$ of the input sequence. The final hidden state representation $(h_1, h_2, h_3, ..., h_n)$ is created by combining the forward and backward representations (often using the concatenation operator). Formally, we have:

$$\overrightarrow{h_t} = LSTM(x_t, \overrightarrow{h}_{t-1}) \tag{8}$$
$$\overleftarrow{h_t} = LSTM(x_t, \overleftarrow{h}_{t-1}) \tag{9}$$
$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}] \tag{10}$$

To make predictions, linear transformations of the hidden states are then passed to a softmax layer.

Similar to the model proposed by Zhang et al. [63], our models use a stack of two Bi-LSTMs. The first Bi-LSTM layer takes the original sequence, $(x_1, x_2, x_3, ..., x_n)$, as input, and produces a hidden state sequence, $h^{(1)}_{1:n} = (h^{(1)}_1, h^{(1)}_2, h^{(1)}_3, ..., h^{(1)}_n)$. The second Bi-LSTM takes the hidden sequence produced by the first Bi-LSTM layer as input, and produces a sequence of deeper hidden state representations $h^{(2)}_{1:n} = (h^{(2)}_1, h^{(2)}_2, h^{(2)}_3, ..., h^{(2)}_n)$. Formally:

$$h^{(1)}_t = BiLSTM_1(x_t, h^{(1)}_{t-1}) \tag{11}$$
$$h^{(2)}_t = BiLSTM_2(h^{(1)}_t, h^{(2)}_{t-1}) \tag{12}$$

Following Zhang et al. [63], we jointly trained the two stacked Bi-LSTMs on two different but related tasks, in a supervised fashion. The first Bi-LSTM was trained to simply find keywords in a text. This can be viewed as a binary classification task, where detected keywords are classified as '1', and other words are classified as '0'. Let the loss function $J_1(\theta)$ represent the classification error of a non-linear transformation of the first hidden layer units, $h^{(1)}_{1:n}$. The hidden layer of the second Bi-LSTM was trained for a relatively more complex task that builds upon the task of the hidden layer of the first Bi-LSTM. Specifically, the second Bi-LSTM was trained to detect keyphrases by using its hidden units $h^{(2)}_{1:n}$ to tag the sequence using the BIOES scheme. In BIOES, 'B' refers to the beginning word of a keyphrase, 'E' refers to the last word of the keyphrase, 'I' refers to the word being inside the keyphrase (in-between 'B' and 'E'), 'S' refers to any single word keyphrase, and 'O' is any word that is not a part of a keyphrase. Let the loss function $J_2(\theta)$ represent the tagging error of a non-linear transformation of the second RNN hidden layer units, $h^{(2)}_{2:n}$. The final loss $J(\theta)$ can then be expressed as the combination of the two losses, $J_1(\theta)$ and $J_2(\theta)$, specifically:

$$J(\theta) = \gamma \cdot J_1(\theta) + (1 - \gamma) \cdot J_2(\theta) \tag{13}$$

where γ is a hyperparameter in the interval [0,1]. The model updates its trainable parameters to minimize the loss function, $J(\theta)$, through backpropagation during training. The loss from a low-level task (keyword discovery) can also serve as additional regularization for the main task (keyphrase extraction). Intuitively, the two stacked networks correspond to two different, but related tasks - essentially, it is an instance of multi-task learning [26, 52].

Contextual Word Embeddings: Word embeddings are high dimensional vector representations of words. Often, embedding algorithms encode contextual information into the vector representations, such that when clustering word-vectors using the Euclidean distance or Cosine-similarity, vectors corresponding to words that appear in similar contexts tend to cluster together. However, classic word embedding models [34, 42] have certain limitations. They can only accommodate one particular vector representation for each word. This restricts the models from being able to capture polysemy (the same word can have completely different meanings in different contexts). The models can also be restricted to a specific vocabulary, which the model has been trained on. Moreover, they can also be blind to morphological information.

Peters et al. [43] proposed the use of a pre-trained language model to create contextualized word embeddings (called ELMo embeddings), which can tackle some of these issues. They used a character level Convolutional Neural Network (CNN) [65] followed by a language model based on Bi-LSTM (however, other architectures can be used [44]). The word embeddings were constructed by a weighted summation of the hidden state representations of the words at various layers of the language model. As a result, there are some interesting benefits. As the initial word-embedding can be done by a CNN at the character level, there is no word-level vocabulary restriction. Also, as the embedding is done by a language model, the embedding of a word is conditioned on the context of all the surrounding words. Therefore, the same word can be represented using different word-vectors if it appears in different contexts.

As an enhancement to the main RNN-based model, which uses Glove embeddings[1] [42] to represent the input words, we concatenated the ELMo embeddings with GloVe embeddings pre-trained on a Twitter corpus. We used Flair[2][2] to load the Twitter GloVe embeddings and TensorFlow hub[3] to load the ELMo embeddings.

Modeling Tweet Noisiness: In Twitter, people often use shorthands, slang, arbitrary capitalizations, and varied lexical representations for a word, all of which contribute to a high overall noisiness

[1] https://nlp.stanford.edu/projects/glove/
[2] https://github.com/zalandoresearch/flair
[3] https://tfhub.dev/google/elmo/2

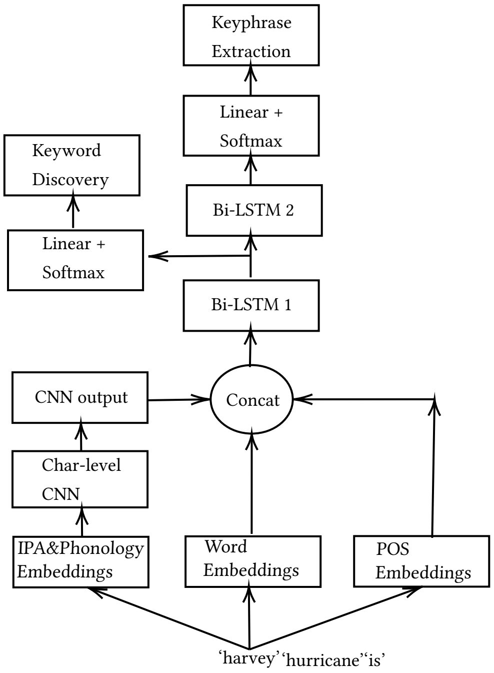


Figure 1: Joint-layer Bi-LSTM with word embeddings, POS-tags embeddings, and IPA & phonological embeddings.

in tweets. For this reason, many Natural Language Processing tools are found to perform noticeably worse when used on Twitter or similar social media domains, even when they perform very well on cleaner data (like news articles) [1, 46]. Aguilar et al. [1] proposed various ways to combat the inherent noisiness in social media platforms. They noted how Twitter users often tend to spell words based on their pronunciations. Thus, while spellings of a word may differ in their lexical format, they tend to have similar phonetics. Therefore, we can create more normalized representations of the words by using their phonetics or corresponding IPA (International Phonetic Alphabet) letters, alongside with their phonological features. Following the work of Aguilar et al. [1], we used Epitran[4] [36] to convert graphemes to phonemes (represented using IPA), and Panphon[5] [37] to convert each IPA phoneme into a vector representing various phonological (articulatory) features associated with it. In addition, we noticed that words that are nouns or adjectives are generally better candidates for keywords. For example, disaster names (nouns) can often be used as keyphrases. Therefore, similar to [1], we also added part-of-speech (POS) information to our model. We used the Twitter optimized POS-tagger[6] as provided by Owoputi et al. [41] for POS-tagging.

We used randomly initialized trainable embeddings for each POS-tag and IPA symbol. We directly used the phonological vector representations created with Panphon as embeddings for phonological features. We concatenated the embeddings of IPA symbols with their corresponding phonological feature vectors. The result of the concatenation, a character level representation of the phonetics and phonological features for each word, was fed to a character-level CNN [65], followed by a global max-pooling layer, to create word level representations. Note that the character-level processing may also allow the network to capture some morphological information. Unlike Aguilar et al., we chose to use a CNN as opposed to a Bi-LSTM for creating the word level representations, because using a Bi-LSTM at character level can be much more computationally expensive and time-consuming than using a CNN. The output of the CNN was concatenated with the POS embeddings, ELMo embeddings, and GloVe embeddings. The resultant representation was then fed to the stacked Bi-LSTM. Figure 1 illustrates this setup.

4 DATASETS

We performed our experiments on both general and disaster-related Twitter datasets. For the general Twitter dataset, we used the dataset[7] published in [63], which is annotated with keyprases. The training and test data were provided as separate files. We used 80% of the training data for actual training and 20% for validation.

We constructed the disaster training and validation datasets by combining disaster-related tweets that have hashtags, from several disasters, specifically, Boston Bombing, Hurricane Harvey, and Hurricane Sandy. We used 80% of the data for training and 20% for validation. We used the combination of disaster-related tweets, with hashtags, from Mexico Earthquake, Hurricane Maria, and Hurricane Irma as test data. To annotate the training, validation and test tweets with keyphrases, we first made use of hashtags. The hashtagged units (marked with '#') in the tweets used are often multi-word phrases without any spaces in-between. We used WordSegment[8] to split the hashtagged multi-word phrases into individual words. These segmented words were used as ground truth for keyphrase extraction. This is similar to how Zhang et al. [63] prepared their annotations. Since hashtags are often used to mark topical phrases or phrases related to some trends, they represent good candidates for keyphrases. In addition, we also used a crisis-lexicon [38] to find disaster-related words (at both unigram and bigram level) in the dataset, and marked them as ground-truth keyphrases. We also added some of our own phrases such as "Boston Bombing", "Hurricane Sandy", "Mexico Earthquake" to the crisis-lexicon.

For both the general and disaster datasets, we filtered out tweets that were longer than 200 tokens or shorter than 5 tokens after tokenization. Table 1 shows the exact number of tweets we used for training, validation, and testing in each dataset. In addition, Table 1 shows the total number of keyphrases and the average number of keyphrases per tweet in each dataset. As can be seen, the average number of keyphrases/hashtags per tweet in the disaster dataset is approximately twice the average number of keyphrases/hashtags per tweet.

There are other differences in terms of keyphrase usage between the general and the disaster datasets. First, the types of keyphrases used in the two datasets are quite different, as the disaster dataset contains a more limited vocabulary related to disasters, while the general dataset has a more diverse vocabulary. This can be seen by comparing the two wordclouds shown in Figure 2. Furthermore, from Figure 2, we can also see that there are many frequently

[4] https://github.com/dmort27/epitran
[5] https://github.com/dmort27/panphon
[6] http://www.cs.cmu.edu/~ark/TweetNLP/
[7] https://github.com/fudannlp16/KeyPhrase-Extraction
[8] http://www.grantjenks.com/docs/wordsegment/

General Twitter	Training Samples	62290
	Validation Samples	15572
	Test Samples:	33392
	Total Keyphrases	111254
	Avg Keyphrases per Tweet	1.00
Disaster Twitter	Training Samples	68128
	Validation Samples	17032
	Test Samples	37106
	Total Keyphrases	221408
	Avg Keyphrases per Tweet	1.81

Table 1: Dataset summary.

occurring keyphrases in the disaster data, whereas there are a lot fewer keyphrases that appear as frequently in the general data. At last, we point out that the disaster test data is quite different from the disaster training (and validation) data, given that the test data contains tweets related to different disasters as compared to the disasters in the training and validation data. In particular, a significant fraction of the disaster test data is made of disaster tweets posted during the Mexico Earthquake, while the training and validation data do not contain tweets related to earthquakes. However, both training/validation and test data contain tweets collected during hurricanes, although the hurricanes used in the training/validation are different from those used in the test. The differences in the disaster training and test data can be observed in Figure 3. We intentionally created a challenging test dataset that allows us to evaluate the model's performance on data from a disaster that does not appear in the training, to understand the usability of the model in a realistic scenario.

5 EXPERIMENTS AND RESULTS

5.1 Experimental Setting

One of our goals was to enhance the original model proposed in [63] with contextual information (in the form of contextual word embeddings), and also with information that can alleviate the noisiness of the Twitter data (specifically, phonology and POS tag information). Thus, our set of experiments was designed to compare the baseline model [63] with the proposed variants. The implementation details of the models are described below. Hyperparameters were selected based on common values used in similar prior architectures, and/or (limited) tuning on the validation set.

Joint-layer Bi-LSTM (J-Bi-LSTM): This model serves as our baseline. We parsed each sequence using a window of size 3, that is, each token in the sequence was represented by three consecutive words. As an example, if the sequence is (s_1, s_2, s_3), after parsing it with a window of size 3, the sequence gets converted into $([SOS, s_1, s_2], [s_1, s_2, s_3], [s_2, s_3, EOS])$ (where SOS is a special tag denoting the start of the sequence, and EOS is a similar tag denoting the end of the sequence). The 3-word representation of a token was used to encode more contextual information into each token, before feeding the tokens into the LSTM. Zhang et al. [63] empirically found 3 to be a good choice for the window size. Each word/token in the sequence was embedded with a 100-dimensional GloVe vector pre-trained on a general Twitter corpus. The embeddings were further fine-tuned during training. Each LSTM network had 300 hidden units. We used a dropout of 0.5 [4] on the input to the first and second LSTMs. We also used a dropout with the same rate on the output from the second LSTM. For training, we used the nadam optimizer [9] with a learning rate of 0.0015. Finally, we used an γ value of 0.5 for joining the two loss functions $J_1(\theta)$ and $J_2(\theta)$. Tuning this hyperparameter may lead to better results.

J-Bi-LSTM with ELMo Embeddings (J-Bi-LSTM+ELMo): This variant is similar to the baseline model, except that it uses 1024 dimensional ELMo embeddings, along with the GloVe embeddings. The two types of embeddings, GloVe and ELMo, are concatenated together. As the ELMo embeddings have much higher dimensionality, and are designed to encode contextual information, we did not use the window approach for this model. Other implementation details and hyperparameters are similar to those in the baseline.

J-Bi-LSTM with IPA & POS (J-Bi-LSTM+IPA+POS): This variant is also similar to the baseline, except that it uses IPA-embeddings, phonological feature vectors and POS-tag embeddings along with the GloVe embeddings (with a window of size 3). We used 64-dimensional embeddings for POS-tags and 22-dimensional embeddings for IPA symbols. The phonological feature vectors have 22 dimensions as well. For the character-level CNN, we used 128 filters and a kernel of size 3. As before, other implementation details and hyperparameters are similar to those in the baseline.

J-Bi-LSTM with ELMo, IPA & POS (J-Bi-LSTM+ELMo+IPA+POS): To evaluate the combined effect of the contextual word embeddings and IPA/POS-tags, this variant makes use of IPA-embeddings, phonological feature vectors and POS-tag embeddings, along with the concatenation of GloVe and ELMo embeddings. Other implementation details and hyperparameters are the same as before.

We compare the above models in three different settings: (1) we train all models on general tweet data and test them also on general tweet data. The goal of this comparison was to see if the additional information used can improve the performance of the baseline; (2) we train all models on the general tweet data and test them on the disaster tweet data, with the goal of understanding how much information can be transferred from the general tweet data to the disaster-specific tweet data; (3) we train all models on disaster tweet data and test them also on disaster tweet data. The goal of this experiment was to see how much the models improve by being trained on specific disaster data, and also what type of information leads to the largest performance improvements.

5.2 Evaluation Metrics

The J-Bi-LSTM based models take a sequence-labeling approach towards keyphrase extraction. Thus, the precision, recall, and F1-measure can be used to evaluate the results. A keyphrase predicted by the model is considered to be correct if there exists an exact matching keyphrase in the same position in the ground-truth annotation; otherwise, the prediction is considered incorrect.

Despite being frequently used, the F1-measure may not be ideal for evaluating the quality of predicted keyphrases. For example, if the ground-truth keyphrase is “harvey”, and the extracted keyphrase is “harvey hurricane”, the exact-F1 measure will be zero. Therefore, the model can produce appropriate predictions, or sometimes even better predictions than the ground-truth annotations, but still, receives a zero score. Similarly, a model may detect some keyphrases that are semantically similar to the ground-truth, but not exact matches. For example, the model may detect ‘tremor’ instead of ‘earthquake’. In these cases too, the model will receive a zero score, whereas presumably some points should be awarded to the model for predicting something close to the ground truth.

The above reasons led us to explore embedding-based measures. It has been shown that the cosine-similarity between certain vector representations (such as GloVe or word2vec embeddings) of semantically similar words tend to be high. Intuitively, we can use cosine-similarity of such word embeddings to measure the degree of similarity between them, as an alternative to simply checking for an exact match. Embedding-based metrics have been used in evaluating the quality of generated dialogues, detecting paraphrases, assessing student responses, and other similar tasks. Here, we are interested in comparing the ground-truth keyphrases with the predicted keyphrases. However, taking the order and positional information into account, while comparing them can be complicated. For example, consider a case where there are two gold keyphrases ‘K_1’ and ‘K_2’ at consecutive locations P_1 and P_2, respectively, whereas the predicted keyphrase is ‘K_1 K_2’ (covering both P_1 and P_2 locations). Such cases make it complicated to use positional information for comparison. Thus, we moved our focus from the standard sequence-labeling evaluation approach to an approach that seems more natural to the task of keyphrase extraction.

Specifically, we first extracted the gold and predicted keyphrases from a labeled sequence and converted them into sets of keyphrases. By doing this, the order information was ignored, and duplicates were removed. As a result, we obtained two sets - a set of gold keyphrases and a set of predicted keyphrases. Our goal was to use the embedding based metrics to measure the overlap between these sets. However, each keyphrase in the gold and predicted sets, respectively, can have an arbitrary number of words, each with its own word embedding. This further complicates the keyphrase-to-keyphrase comparison. To simplify the task, we took the average of the word-embeddings in each phrase to create phrase-level embeddings. Given the phrase-level embeddings, the embedding-based metrics can be used in several ways to score the overall quality of the predicted keyphrases using the gold keyphrases as the standard. Some of these methods are described below.

Average Embeddings: One approach is to take the average [11, 22, 35] of all phrase-level embeddings in the set of gold keyphrases. (Alternatively, we can also use the most extreme value [12] along each dimension, across all the phrase-level-embeddings in the set.) Then, we can either compute the average of cosine-similarity scores between the phrase-level embedding of each predicted keyphrase and the average-gold-embedding, or we can compute the cosine-similarity score between the average embedding of the predicted keyphrases and the average embedding of the gold keyphrases.

Greedy Matching: Another approach is to make word to word (or in our case, keyphrase to keyphrase) comparisons without creating single representations for all keyphrases in the sets. In this approach, we can score each keyphrase from the set of predicted keyphrases in two steps. In the first step, we compute the cosine-similarity scores between the predicted keyphrase under consideration and each keyphrase in the set of gold keyphrases. In the second step, we take the maximum of all the computed cosine-similarity scores for each predicted keyphrase. The maximum cosine-similarity score should correspond to the best matching keyphrase in the gold set for a particular predicted keyphrase. Thus, each predicted keyphrase can be scored. Finally, we can take the average of the scores to get the final score for the model’s prediction [48]:

$$GM(\,pred, gold\,) = \frac{\sum_{i=1}^{p} score(pred_i, gold)}{p} \tag{14}$$

where p is the number of predicted keyphrases, *pred* is the set of predicted keyphrases, *gold* is the set of gold keyphrases, and $pred_i$, with $i \in \overline{1,p}$, are the elements of *pred*. *GM* is the greedy match scoring function, and $score(pred_i, gold)$ is the maximum cosine-similarity achieved between $pred_i$ and any keyphrase from *gold*.

However, this scoring is not symmetric. $GM(\,pred, gold\,)$ will not necessarily be the same as $GM(\,gold, pred\,)$. Inspired from [48], we can use the following formula to calculate a symmetric score:

$$T = GM(\,pred, gold\,) + GM(\,gold, pred\,) \tag{15}$$

$$SymmGM(\,pred, gold\,) = \frac{T}{2} \tag{16}$$

There is another benefit of the symmetric scoring. If $GM(pred, gold)$ is used alone, the model can be scored highly even if there is only one prediction in the set which happens to match very well with just one gold keyphrase, among multiple other gold keyphrases that were not predicted. Similarly, $GM(\,gold, pred\,)$ can produce a high score, even if there is only one keyprhase in the gold set that matches very well with just one predicted keyphrase, among multiple other predicted keyphrases that do not match with the gold. Using $SymmGM(\,pred, gold\,)$ mitigates these issues.

Optimal matching: This approach, proposed by Rus and Lintean [48], can be used to first create an optimal group of exclusive phrase-to-phrase pairs using the Kuhn-Munkres method [20] for optimal assignment. Then, each predicted keyphrase can be scored simply by computing its cosine-similarity with the gold-keyphrase that it is paired with. The later steps are similar to those of greedy matching.

Proposed embedding-based metric: We chose greedy matching as the backbone of our proposed embedding-based metric. As we are dealing with small sets of discrete keyphrases, we found it more appropriate to choose a phrase-to-phrase comparison approach such as greedy matching or optimal matching. A predicted keyphrase can be considered appropriate if it is similar to any of the gold keyphrases. Therefore, to score the correctness of the predictions, we chose to allow multiple predicted keyphrases to be matched to the same gold keyphrase if needed. However, optimal matching can only allow exclusive one-to-one pairs. This is why we did not choose the optimal matching.

We should note that if we simply use the average score when calculating $GM(\,pred, gold\,)$ and $GM(\,gold, pred\,)$, the variation in the

	Gold and prediction	F1	No alpha-beta No threshold General Embd	No alpha-beta No threshold Crisis Embd	General Embd	Crisis Embd
1	**Gold:** teen choice **Prediction:** teen choice awards	0%	95.6%	88.4%	95.6%	88.4%
2	**Gold:** earthquake, volunteers, working together **Prediction:** earthquake, volunteers working together	40%	93.1%	92.1%	80.5%	79.8%
3	**Gold:** storm, cloud, calamity **Prediction:** tornado, cloudy sky, disaster	0%	61.1%	30.9%	52.3%	13.4%
4	**Gold:** harvey **Prediction:** hurricane	0%	25.2%	63.20%	0%	63.20%
5	**Gold:** hurricane harvey, hurricane, katrina, red cross, bombing **Prediction:** hurricane, harvey, katrina hurricane, cross, boston bombing	20%	90%	93.4%	90%	93.4%
6	**Gold:** boston bombing, hurricane harvey, red cross **Prediction:** hurricane harvey	50%	83.3%	77.4%	54.2%	44.4%
7	**Gold:** boston bombing, hurricane harvey **Prediction:** hurricane harvey, boston, bombing, red cross	33.3%	90.2%	90.3%	70.3%	70.2%

Table 2: General and crisis embedding-based scores versus F1 scores on seven sets of gold/predicted keyphrases.

number of keyphrases in *gold* and *pred* may not be completely taken into account by greedy matching. More specifically, $GM(pred, gold)$ can be high as long as all the predicted keyphrases achieve high cosine-similarity scores with any one of the gold keyphrases. But it may not take into account how many more or less keyphrases there may be in the *gold* set as compared to those in the *pred* set. Similarly, $GM(gold, pred)$ can be high, as long as all the gold keyphrases achieve high cosine-similarity scores with any of the predicted keyphrases, without taking into account how many more or less keyphrases there may be in the *pred* set as compared to those in the *gold* set. Both $GM(pred, gold)$ and $GM(gold, pred)$ may be high at the same time despite wide variation in the number of predicted keyphrases and gold keyphrases, when most of the keyphrases in *gold* and *pred* are very similar to each other. To account for this, we may want to penalize the model for extracting too little or too many keyphrases compared to the gold. This can be done with a simple modification to the *GM* definition:

$$GM'(pred, gold) = \frac{\sum_{i=1}^{p} score(pred_i, gold)}{max(p, g)} \tag{17}$$

$$GM'(gold, pred) = \frac{\sum_{i=1}^{g} score(gold_i, pred)}{max(p, g)} \tag{18}$$

$$T' = GM'(pred, gold) + GM'(gold, pred) \tag{19}$$

$$SymmGM'(pred, gold) = \frac{T'}{2} \tag{20}$$

where g is the number of gold keyphrases, and p is the number of predicted keyphrases as defined before. $SymmGM'(pred, gold)$ is the overall score. If g is higher than p, the $GM'(pred, gold)$ score will be lower as compared to the $GM(pred, gold)$ score, and if p is higher than g, then $GM'(gold, pred)$ will be lower as compared to $GM(gold, pred)$. Intuitively, we can think of $GM'(pred, gold)$ as penalizing the overall score, when g is higher than p, and we can think of $GM'(gold, pred)$ as penalizing the overall score when p is higher than g.

However, we may also want to have more fine-grained control over the degree of penalty. Typically, $SymmGM(pred, gold)$ will be high even when there is wide variation in the lengths of *pred* and *gold* sets if most of the predicted and gold kephrases are similar to each other. But if that is the case, the predictions are still likely to be appropriate. Moreover, in general, variation in the number of keyphrases can be implicitly penalized by plain averages (with $GM(gold, pred)$ and $GM(pred, gold)$) because with wide variation, there will be higher chances for there to be phrases unique to only one set, which are dissimilar with any of the phrases in the other set. For these reasons, we may want to constrain the penalty for the variation of length, which can be achieved as another extension of the *GM* scoring. First note that:

$$max(p, g) = p + max(0, g - p) = g + max(0, p - g) \tag{21}$$

To gain more fine-grained control over the penalty for the length variation, we use two new parameters α and β, as follows:

$$GM_{ext}(pred, gold) = \frac{\sum_{i=1}^{p} score(pred_i, gold)}{p + \alpha \cdot max(0, g - p)} \tag{22}$$

$$GM_{ext}(gold, pred) = \frac{\sum_{i=1}^{g} score(gold_i, pred)}{g + \beta \cdot max(0, p - g)} \tag{23}$$

$$T_{ext} = GM_{ext}(pred, gold) + GM_{ext}(gold, pred) \tag{24}$$

$$G(pred, gold) = \frac{T_{ext}}{2} \tag{25}$$

Intuitively, α and β can be used to modulate the penalty for p being smaller than g, and for p being greater than g, respectively. This leads to a more general formulation. For example, when α is 0, $GM_{ext}(pred, gold)$ is just $GM(pred, gold)$, and when α is 1, $GM_{ext}(pred, gold)$ is just $GM'(pred, gold)$. $G(pred, gold)$ is symmetric when α and β have the same value.

We used $G(pred, gold)$ to compare a predicted keyphrase set with a gold keyphrase set, with α and β both being set to 0.7. In addition, we also used a threshold θ of 0.4 on each cosine-similarity scores to limit the contribution of lower-scoring matches, and implicitly, to limit the range of $G(pred, gold)$ to values between 0 and 1. That is, if the cosine similarity score is lower than a specified threshold, we assigned it a score of 0, otherwise, the actual score is used. The values of α, β, and θ were selected intuitively. There is room for empirical studies in the future to determine how people judge the evaluation quality for different values of α, β, and θ.

Embedding-based versus F1-measure evaluation: In Table 2, we used several sets of predicted keyphrases and gold keyphrases to compare the exact-match F1-measure with greedy matching (no alpha-beta and no threshold), and our proposed embedding-based metric. The last five examples are fabricated pairs of gold keyphrases and predictions to serve as case studies for comparison. For the embedding-based metric, we used both GloVe embeddings pre-trained on general Twitter (General Embd), and GloVe embeddings that we trained on a crisis-related Twitter corpus (Crisis Embd). The crisis-related embeddings were trained on three previously published crisis datasets, specifically, CrisisLexT6 [39], CrisisLexT26 [40] and 2CTweets [50], together with tweets collected using the Twitter Streaming API during Hurricanes Harvey, Irma, and Maria, and Mexico Earthquake. The total corpus contains approximately 5.8 million tweets. As can be seen in Table 2, the embedding-based scores appear to be more appropriate than the F1 scores, based on a human understanding of the keyphrases. We can also notice (see example 3) that the general Twitter GloVe embeddings can be better for scoring the semantic similarity between general words than the crisis embeddings. This may be because the general Twitter embeddings were trained on a much larger corpus, which is more diverse and general. However, crisis embeddings appear to be better for scoring the similarity between disaster-related words, especially if disaster-related named-entities are involved (see example 4). This is probably because the crisis embeddings were trained on more recent Twitter data and, specifically, on disaster-related data. Not so surprisingly, we also find that crisis embedding-based scores are higher on the disaster data, and general Twitter embedding-based scores are higher on the general data in Table 4. Alpha-beta modulated embedding scores are generally more stringent than embedding scores without the modulation. Without alpha-beta modulation and thresholding, the embedding metrics work as standard greedy matching. The difference made by alpha-beta modulation is most sharply noticeable in examples 6 and 7 where there is a significant difference in the number of gold keyphrases and predicted keyphrases. As can be seen, without alpha-beta modulation, the embedding scores are very high despite the difference in length between the two sets of keyphrases.

5.3 Results

The results of our experiments for the three settings discussed in Section 5.1 are presented in Tables 3 and 4. Table 3 shows precision, recall, and F1 scores, based on exact keyphrase matches (where the order of the keyphrases and the position are both taken into account). Table 4 shows F1 scores versus GM_{ext} embedding-based scores, computed by comparing a set of predicted keyphrases with a set of gold keyphrases (where the order of the keyphrases and the position are both ignored, and duplicate keyphrases are removed). Thus, while the F1 scores in Tables 3 and 4 may appear to be inconsistent, we would like to point out that the differences are due to the way the scores are calculated in each table, as explained above.

Model	**P**	**R**	**F_1**
Trained and tested on general data			
J-Bi-LSTM	83.95%	80.81%	82.35%
J-Bi-LSTM+ELMo	84.39%	**81.65%**	82.99%
J-Bi-LSTM+IPA+POS	84.94%	81.63%	83.25%
J-Bi-LSTM+ELMo+IPA+POS	**85.25%**	81.53%	**83.35%**
Trained on general data; tested on disaster data			
J-Bi-LSTM	**37.49%**	**09.92%**	**15.68%**
J-Bi-LSTM+ELMo	35.88%	09.69%	15.26%
J-Bi-LSTM+IPA+POS	34.53%	08.79%	14.01%
J-Bi-LSTM+ELMo+IPA+POS	36.50%	09.06%	14.51%
Trained and tested on disaster data			
J-Bi-LSTM	65.21%	61.46%	63.28%
J-Bi-LSTM+ELMo	67.67%	59.60%	63.38%
J-Bi-LSTM+IPA+POS	**69.06%**	**61.98%**	**65.33%**
J-Bi-LSTM+ELMo+IPA+POS	67.14%	60.37%	63.57%

Table 3: Precision, Recall, and F1 scores based on the sequence-labeling evaluation (using exact matches).

Model	**F1**	**General Embd**	**Crisis Embd**
Trained and tested on general data			
J-Bi-LSTM	84.26%	92.45%	89.46%
J-Bi-LSTM+ELMo	84.41%	92.64	89.67%
J-Bi-LSTM+IPA+POS	84.79%	92.88%	90.06%
J-Bi-LSTM+ELMo+IPA+POS	**85.10%**	**93.22%**	**90.42%**
Trained on general data; tested on disaster data			
J-Bi-LSTM	**20.45%**	32.64%	44.98%
J-Bi-LSTM+ELMo	19.38%	32.74%	45.89%
J-Bi-LSTM+IPA+POS	19.11%	34.33%	**46.49%**
J-Bi-LSTM+ELMo+IPA+POS	20.28%	**34.34%**	45.77%
Trained and tested on disaster data			
J-Bi-LSTM	64.05%	78.05%	82.03%
J-Bi-LSTM+ELMo	64.12%	78.41%	**82.51%**
J-Bi-LSTM+IPA+POS	**66.37%**	**78.55%**	82.40%
J-Bi-LSTM+ELMo+IPA+POS	65.11%	78.16%	82.25%

Table 4: General and crisis embedding-based scores versus F1 scores, when comparing predicted with gold keyphrases.

We calculated the F1-scores in Table 4 by comparing sets, in order to make them more comparable to the embedding-based scores. We will next analyze the performance of each model individually.

Baseline (J-Bi-LSTM): The baseline achieved scores in the low-to-mid 80% range when trained and tested on general data, and scores in the low-to-mid 60% range when trained and tested on disaster data. Surprisingly, its performance is close to the performance of other models, which use richer information. Furthermore, the J-Bi-LSTM model produced higher recall than models with ELMo on disaster test data. Also, interestingly, the baseline model achieved the best F1 score in Table 3, when trained on general data and tested on disaster data. However, it was not the best scorer if we consider its embedding-based scores in Table 4 instead.

J-Bi-LSTM+ELMo: The model enhanced with contextual word embeddings achieved a slightly higher score than the baseline when trained and tested on general Twitter data. An even better score may have been possible if the ELMo embeddings were pre-trained on a Twitter corpus. However, the F1 performance of this model on disaster data after being trained on general data is lower than the performance of the baseline, but its embedding-based scores are higher. There is no significant difference between the overall F1 score of the baseline and the F1 score of this model when trained and tested on disaster data. Although the model seems to get slightly better precision, it struggles on recall more than the baseline.

J-Bi-LSTM+IPA+POS: The addition of POS, IPA and phonological feature information generally improves the performance of the J-Bi-LSTM+IPA+POS model over the baseline. The model even gets a better score than the J-Bi-LSTM+ELMo model, and it achieves similar performance to J-Bi-LSTM+ELMo+IPA+POS on the general test data. So, simply adding POS, IPA and phonological information may eliminate the need for ELMo embeddings. This could be because both ELMo and IPA (with phonological features) can model some similar character-level information (like morphology). Furthermore, as mentioned above, ELMo may not be expressing its full potential in the Twitter domain because it was not pre-trained on a noisy social media corpus. Interestingly, the J-Bi-LSTM+IPA+POS appears to have the worst F1 score when trained on general data and tested on disaster data, but it performs the best in terms of the crisis embedding-based metric. When trained and tested on disaster data, this model is also one of the better performers.

J-Bi-LSTM+ELMo+IPA+POS: Finally, the model enhanced with ELMo embeddings, and also POS, IPA and phonological feature information, achieved the best overall F1 score when trained and tested on general data, but not in the other two scenarios. It seems that the ELMo embeddings and the POS/phonetics may contain somewhat overlapping information. ELMo can capture some character level information, which may be why adding further character level information from IPA and phonological features do not give a significant boost in performance. Overall the performance of this model is similar to the performance of the JRNN+ELMo model on the disaster data. We also considered training models using crisis embeddings but they did not help.

In the remaining of this section, we will use the results in Tables 3 and 4 to answer our original questions. First, regarding question (1), we can say that the models enhanced with contextual and/or POS/phenomes information generally perform better than the baseline model, although the baseline is not much worse.

Regarding our original question (2), we noted that the models achieved much lower F1 scores when trained on general data and tested on disaster data. It should be expected that the domain differences contribute to the low score. However, there is another important reason. As we mentioned before in Section 4, the average number of keyphrases per disaster tweet is approximately twice the average number of keyphrases per general tweet. Thus, it is likely that the models, when trained on the general tweet data, learn to predict one keyphrase per sample on average. This results in a substantial hit to their recall when tested on the disaster data, which presents more variety in the number of keyphrases per tweet. Nevertheless, as can be seen from figure 4(a), a model trained on general data, can still predict keyphrases similar to the gold (Figure 3(b)). Furthermore, their embedding-based scores are also much higher than their overall F1 scores. Due to the fundamental differences in the two datasets (as discussed in more details in Section 4), a model may learn some parameters which are useful for general tweets, but the same parameters may be counter-productive in the case of tweets from the disaster test data. This may be why the models that performed better on general data struggled more to generalize on disaster data. But the results are not too straightforward, because some of those models did generalize better on disaster data when considering their embedding-based scores. Overall, it may be better to be cautious about judging the models' capabilities based on their performance on a domain they were not even trained on.

Finally, to answer question (3), we can claim that the models trained and tested on the disaster data can perform significantly better than the models trained on general data and tested on disaster data. However, the performance scores on the disaster test data are overall lower than those on the general test data. This is because we used a rather challenging test set for the disaster data, as discussed earlier. Furthermore, the low diversity of keyphrases in the disaster training data can encourage the models to memorize more (and thus overfit) instead of learning more general abstract features. Nevertheless, they can still predict keyphrases that are fairly similar to the gold (see Figures 3(b) and 4(b)), and their embedding-based scores are better than the F1 scores.

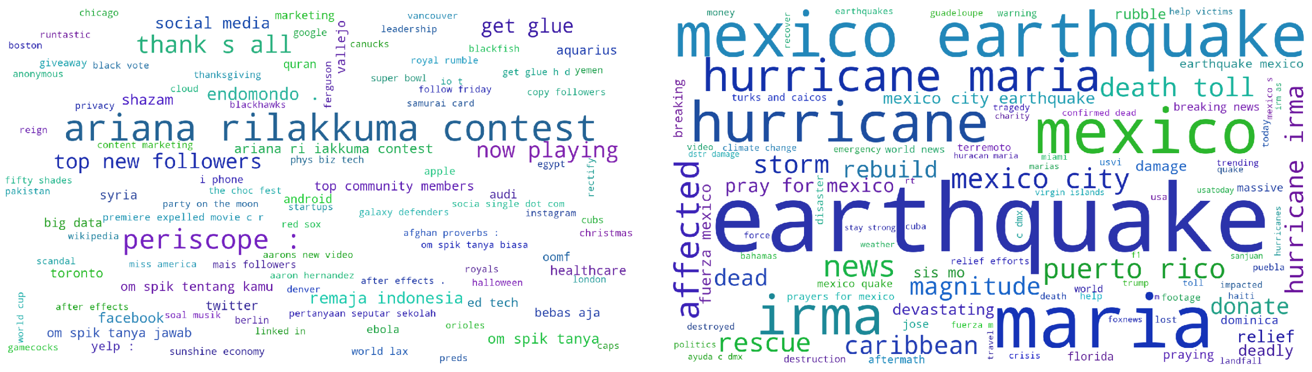
6 CONCLUSIONS AND FUTURE WORK

We showed that the Joint-layer Bi-LSTM model can be a promising approach for extracting disaster-related keyphrases from Tweets. We also showed that ELMo embeddings and/or phonetics and phonological features, along with POS-tags, can generally improve the performance of the model. We discussed how the exact match F1 can be an unsuitable metric for evaluating the quality of the predicted keyphrases, and we showed that embedding-based metrics are more appropriate for this task. We further introduced novel extensions to the embedding-based metrics, to account for differences in terms of the number of predicted and gold keyphrases.

In future work, we would like to focus on abstractive keyphrase generation [32]. Taking a generative approach as opposed to an extractive one would allow us to generate keyphrases that are absent from the source text but still appropriate given the semantics of the text. This can further enhance keyphrase-based search of tweets related to a particular context (including disaster context).

ACKNOWLEDGEMENTS

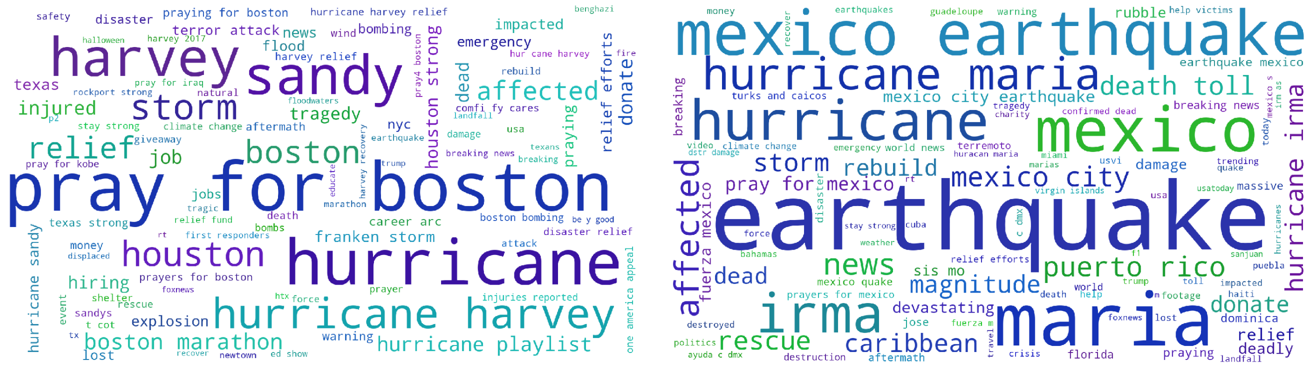
We thank the National Science Foundation (NSF) and Amazon Web Services for support from grants IIS-1741345 and IIS-1912887, which supported the research and the computation in this study. We also thank NSF for support from the grants IIS-1526542, IIS-1423337, IIS-1652674, and CMMI-1541155.



(a) General test data

(b) Disaster test data

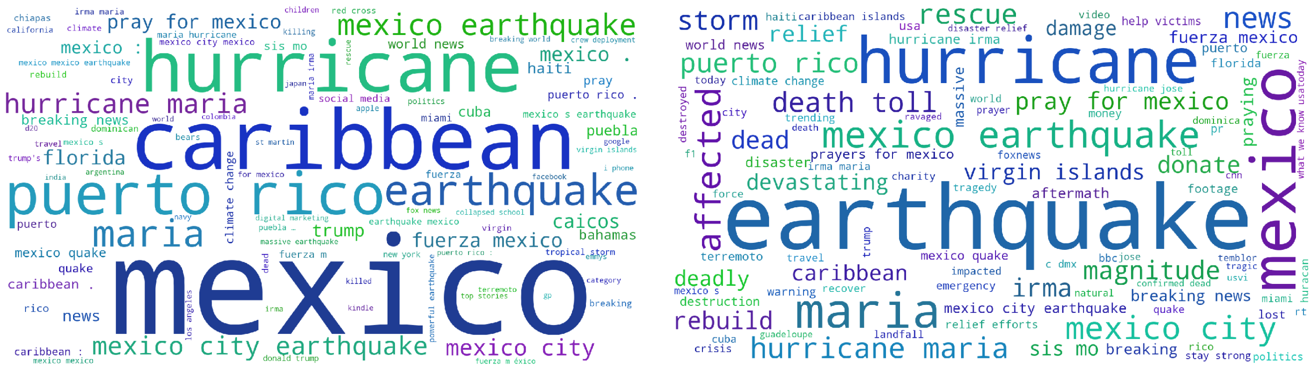
Figure 2: Word clouds of the top 100 most frequent gold keyphrases from the general test data (a) and disaster test data (b).



(a) Disaster training data

(b) Disaster test data

Figure 3: Word clouds of the top 100 most frequent gold keyphrases from disaster training data (a) and disaster test data (b).



(a) Predictions by model trained on general data

(b) Predictions by model trained on disaster data

Figure 4: Word clouds of the top 100 most frequent keyphrases from the disaster test data, as predicted by the J-Bi-LSTM+IPA+POS model trained on the general data (a) and J-Bi-LSTM+IPA+POS trained on disaster data (b).

REFERENCES

[1] Gustavo Aguilar, Adrian Pastor López Monroy, Fabio González, and Thamar Solorio. 2018. Modeling Noisiness to Recognize Named Entities using Multitask Neural Networks on Social Media. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Vol. 1. Association for Computational Linguistics, New Orleans, Louisiana, 1401–1412.

[2] Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual String Embeddings for Sequence Labeling. In *COLING*. Association for Computational Linguistics, Santa Fe, New Mexico, USA, 1638–1649.

[3] Zahra Ashktorab, Christopher Brown, Manojit Nandi, and Aron Culotta. 2014. Tweedr: Mining twitter to inform disaster response. In *ISCRAM*.

[4] Pierre Baldi and Peter J Sadowski. 2013. Understanding dropout. In *Advances in neural information processing systems*. 2814–2822.

[5] Abdelghani Bellaachia and Mohammed Al-Dhelaan. 2012. Ne-rank: A novel graph-based keyphrase extraction in twitter. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01*. IEEE Computer Society, 372–379.

[6] Nicola Bruno. 2011. Tweet first, verify later? How real-time information is changing the coverage of worldwide crisis events. *Reuters Institute for the Study of Journalism* (2011), 2010–2011.

[7] Cornelia Caragea, Florin Adrian Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. 2014. Citation-Enhanced Keyphrase Extraction from Research Papers: A Supervised Approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. 1435–1446.

[8] Guy Paul Cooper Jr, Violet Yeager, Frederick M Burkle Jr, and Italo Subbarao. 2015. Twitter as a potential disaster risk reduction tool. Part I: Introduction, terminology, research and operational applications. *PLoS currents* 7 (2015).

[9] Timothy Dozat. 2016. Incorporating nesterov momentum into adam. (2016).

[10] Corina Florescu and Cornelia Caragea. 2017. PositionRank: An Unsupervised Approach to Keyphrase Extraction from Scholarly Documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. 1105–1115.

[11] Peter W Foltz, Walter Kintsch, and Thomas K Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse processes* 25, 2-3 (1998), 285–307.

[12] Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *Nips, modern machine learning and natural language processing workshop*, Vol. 2.

[13] Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-Manning. 1999. Domain-Specific Keyphrase Extraction. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI '99)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 668–673. http://dl.acm.org/citation.cfm?id=646307.687591

[14] Willa Frej. 2018. Hurricane Florence Flood Victims Turn To Social Media For Rescue. *HuffPost* (2018). https://www.huffingtonpost.com/entry/hurricane-florence-flood-victims-social-media_us_5b9b86c0e4b013b097799cd1

[15] Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting Keyphrases from Research Papers Using Citation Networks. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.* 1629–1635.

[16] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *Automatic Speech Recognition and Understanding*. IEEE, 273–278.

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

[18] Anette Hulth. 2003. Improved Automatic Keyword Extraction Given More Linguistic Knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP '03)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 216–223. https://doi.org/10.3115/1119355.1119383

[19] Muhammad Imran, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. 2015. Processing social media messages in mass emergency: A survey. *ACM Computing Surveys (CSUR)* 47, 4 (2015), 67.

[20] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97.

[21] Shamanth Kumar, Geoffrey Barbier, Mohammad Ali Abbasi, and Huan Liu. 2011. TweetTracker: An Analysis Tool for Humanitarian and Disaster Relief.. In *ICWSM*.

[22] Thomas K Landauer and Susan T Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review* 104, 2 (1997), 211.

[23] Tamar Lapin. 2018. Family's tweet for help leads to rescue during Hurricane Florence. *New York Post* (2018). https://nypost.com/2018/09/16/familys-tweet-for-help-leads-to-rescue-during-hurricane-florence/

[24] Marina Litvak, Mark Last, Hen Aizenman, Inbal Gobits, and Abraham Kandel. 2011. DegExt—A language-independent graph-based keyphrase extractor. In *Advances in Intelligent Web Mastering–3*. Springer, 121–130.

[25] Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023* (2016).

[26] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101* (2016).

[27] Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*. Association for Computational Linguistics, 257–266.

[28] Patrice Lopez and Laurent Romary. 2010. HUMB: Automatic Key Term Extraction from Scientific Articles in GROBID. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval '10)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 248–251. http://dl.acm.org/citation.cfm?id=1859664.1859719

[29] Douglas MacMillan. 2017. In Irma, Emergency Responders' New Tools: Twitter and Facebook. *The Wall Street Journal* (2017). https://www.wsj.com/articles/for-hurricane-irma-information-officials-post-on-social-media-1505149661

[30] Luis Marujo, Wang Ling, Isabel Trancoso, Chris Dyer, Alan W Black, Anatole Gershman, David Martins de Matos, Joao Neto, and Jaime Carbonell. 2015. Automatic keyword extraction on twitter. In *IJCNLP*. 637–643.

[31] Olena Medelyan, Eibe Frank, and Ian H Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *EMNLP*. 1318–1327.

[32] Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. *arXiv preprint arXiv:1704.06879* (2017).

[33] Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

[34] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013). arXiv:1301.3781 http://arxiv.org/abs/1301.3781

[35] Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. *proceedings of ACL-08: HLT* (2008), 236–244.

[36] David R. Mortensen, Siddharth Dalmia, and Patrick Littell. 2018. Epitran: Precision G2P for Many Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)* (7–12), Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga (Eds.). European Language Resources Association (ELRA), Paris, France.

[37] David R. Mortensen, Patrick Littell, Akash Bharadwaj, Kartik Goyal, Chris Dyer, and Lori S. Levin. 2016. PanPhon: A Resource for Mapping IPA Segments to Articulatory Feature Vectors. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. ACL, 3475–3484.

[38] Alexandra Olteanu, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. 2014. CrisisLex: A Lexicon for Collecting and Filtering Microblogged Communications in Crises. In *ICWSM*.

[39] Alexandra Olteanu, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. 2014. CrisisLex: A Lexicon for Collecting and Filtering Microblogged Communications in Crises. In *Proceedings of the Eighth International Conference on Weblogs and Social Media, ICWSM 2014, Ann Arbor, Michigan, USA, June 1-4, 2014.* http://www.aaai.org/ocs/index.php/ICWSM/ICWSM14/paper/view/8091

[40] Alexandra Olteanu, Sarah Vieweg, and Carlos Castillo. 2015. What to Expect When the Unexpected Happens: Social Media Communications Across Crises. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work &#38; Social Computing (CSCW '15)*. ACM, New York, NY, USA, 994–1009. https://doi.org/10.1145/2675133.2675242

[41] Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: human language technologies*. 380–390.

[42] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[43] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* (2018).

[44] Matthew E Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018. Dissecting Contextual Word Embeddings: Architecture and Representation. *arXiv preprint arXiv:1808.08949* (2018).

[45] Maya Rhodan. 2017. 'Please Send Help.' Hurricane Harvey Victims Turn to Twitter and Facebook. *Time* (2017). http://time.com/4921961/hurricane-harvey-twitter-facebook-social-media/

[46] Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, 1524–1534.

[47] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory* (2010), 1–20.

[48] Vasile Rus and Mihai Lintean. 2012. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*. Association for Computational Linguistics, 157–162.

[49] Gerard Salton and Michael J. McGill. 1986. *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc., New York, NY, USA.

[50] Axel Schulz, Christian Guckelsberger, and Frederik Janssen. 2017. Semantic Abstraction for generalization of tweet classification: An evaluation of incident-related tweets. *Semantic Web* 8, 3 (2017), 353–372.

[51] Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[52] Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vol. 2. 231–235.

[53] Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* 28, 1 (1972), 11–21.

[54] Kevin Stowe, Michael J Paul, Martha Palmer, Leysia Palen, and Kenneth Anderson. 2016. Identifying and categorizing disaster-related tweets. In *Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media*. 1–6.

[55] Tai Tran. 2016. Twitter's 10th Birthday: The Power Of Real-Time Content And Recount Of Memorable Tweets. *Forbes* (2016). https://www.forbes.com/sites/taitran/2016/03/21/twitters-10th-birthday-the-power-of-real-time-content-recount-of-memorable-tweets/

[56] Peter D. Turney. 1999. Learning to Extract Keyphrases from Text. *CoRR* cs.LG/0212013 (1999).

[57] Peter D. Turney. 2000. Learning Algorithms for Keyphrase Extraction. *Inf. Retr.* 2, 4 (May 2000), 303–336. https://doi.org/10.1023/A:1009976227802

[58] Sudha Verma, Sarah Vieweg, William J Corvey, Leysia Palen, James H Martin, Martha Palmer, Aaron Schram, and Kenneth Mark Anderson. 2011. Natural Language Processing to the Rescue? Extracting" Situational Awareness" Tweets During Mass Emergency.. In *ICWSM*. Barcelona, 385–392.

[59] Sarah Vieweg, Amanda L Hughes, Kate Starbird, and Leysia Palen. 2010. 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*. ACM, 1079–1088.

[60] Carlos Villegas, Matthew Martinez, and Matthew Krause. 2018. Lessons from Harvey: Crisis Informatics for Urban Resilience. *Rice University Kinder Institute for Urban Research* (2018).

[61] Awen Wen. 2017. How Social Media is Revolutionizing Real-time Information in Six Ways. *Medium* (2017). https://medium.com/@awenz/how-social-media-is-revolutionizing-real-time-information-in-six-ways-335b3a1624ed

[62] Ian H Witten and Olena Medelyan. 2006. Thesaurus based automatic keyphrase indexing. In *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'06)*. IEEE, 296–297.

[63] Qi Zhang, Yang Wang, Yeyun Gong, and Xuanjing Huang. 2016. Keyphrase extraction using deep recurrent neural networks on Twitter. In *EMNLP*. 836–845.

[64] Shanshan Zhang and Slobodan Vucetic. 2016. Semi-supervised discovery of informative tweets during the emerging disasters. *arXiv preprint arXiv:1610.03750* (2016).

[65] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*. 649–657.

[66] Wayne Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achananuparp, Ee-Peng Lim, and Xiaoming Li. 2011. Topical keyphrase extraction from twitter. In *ACL*. Association for Computational Linguistics, 379–388.