Disaster-Related Tweet Classification

Jishnu Ray Chowdhury
jraych2@uic.edu

Krishna Garg
kgarg8@uic.edu

Mobashir Sadat
msadat3@uic.edu

Chirag Chhablani
cchhab2@uic.edu

Deblina Roy
droy7@uic.edu

Akshat Pancholi
apanch21@uic.edu

Abstract

Social Media platforms like Twitter are important sources of information during disasters. Exploiting these platforms can facilitate disaster management and potentially help save more lives during times of disaster. To achieve this goal, it is necessary to filter disaster-related information from the endless streams of data that are constantly being generated in popular Social Media platforms every minute. In this project, we tackled the task of classifying Disaster-related Tweets from a Twitter Dataset. We developed and compared different BERT-based [9] model variants for both binary classification and multi-class classification under a multi-task learning framework. Precisely, we explored attention-based and capsule routing-based aggregation methods to aggregate Bi-LSTM [18, 13] hidden states and BERT layers to improve the classification performance. We discover that the simple BERT-only baseline is surprisingly competitive and the more sophisticated variants do not help as much. In general, we achieve high performance - more than 80% in different evaluation metrics (F1, accuracy) for both multi-class classification and binary classification. We make our code publicly available in Github[1].

1 Introduction

During disasters, the affected individuals often turn to social media platforms such as Twitter and Facebook to update their status or to get the latest updates about warnings, announcements, and rescue operations. Some victims even use Tweets to request for help, food, water, and supplies in crises. Traditional news sources cannot keep up with the much easily accessible real-time sources like Twitter, Facebook, Reddit where anyone can post their own 'news'. Nonetheless, the value of the information posted on social media platforms during disasters is highly unexploited, partly due to the lack of tools that can help filter the relevant, informative, and actionable messages. Thus, in this project, we focused on the classification of disaster-related tweets so that it is possible to effectively filter out unrelated data. While there are several prior works [30, 31, 22, 20, 21, 29, 3, 1, 2, 26] that focus on Disaster classification, as noted by Ma et al. [28], there is no standard large scale dataset with clear baselines. Ma et al. [28] took the first step in aggregating multiple datasets for Disaster classification and establishes some clear baselines with modern high performing NLP models like BERT [9]. Our works build upon the works of Ma et al. [28]. We only compared BERT-based models since Ma et al. [28] showed that BERT brings significant improvement over baselines that do not use BERT-like pre-trained language models.

[1] https://github.com/JRC1995/BERT-Disaster-Classification-Capsule-Routing

Our main contributions can be summarized as follows:

1. We empirically investigated the effect of using attention-based layer aggregation of the last few layers of BERT.
2. We empirically investigated the effect of using capsule-routing-based layer aggregation of the last few layers of BERT.
3. We empirically investigated the effect of using attention-based aggregated Bi-LSTM hidden states (instead of just using the final hidden states) for classification.
4. We empirically investigate the effect of using capsule-routing-based aggregated Bi-LSTM hidden states (instead of just using the final hidden states) for classification.
5. We setup a multi-task learning framework so that our model can utilize the extra data available for binary classification while also learning multi-class classification.
6. We collected and aggregated Twitter classification datasets from various sources. We relabeled them to maintain consistency and we make our aggregated dataset publicly available on Github.

2 Related Work

There are many previous works related to processing social media during emergency situations [19, 20, 21, 34], extracting information from Twitter to enhance situational awareness [39, 40, 25], and classifying or clustering disaster-relevant tweets [4, 47, 36, 19, 20, 21, 30, 31, 22, 29, 3, 1, 2, 26, 28], among others.

Using an attention mechanism [5] after a Bi-LSTM [18, 13] to aggregate its hidden states just before classification [48] is a standard practice used in multiple previous works. Ma et al. [28] used a BERT-BiLSTM stack similar to us but they did not try attention mechanism-based hidden state aggregation unlike us.

Using layer-aggregation is relatively rare and as far as we are aware, no previous works explored layer-aggregation with BERT for classification as we do. However, there are some previous line of works on layer aggregation on LSTMs and Transformers [38]. ELMo [33] showed the benefit of using a weighted summation of different layers instead of just using the last layer for encoding. There had been other explorations with various more sophisticated ways (for example, using a convolution network or vector weights instead of simple scalar-weighted summation) for layer aggregation particularly in computer vision and machine translation [10, 11, 45, 41]. The most recent approaches [11] use capsule routing [16, 35, 17] for layer aggregation.

Although capsule networks [16, 35, 17] were originally introduced in computer vision to address the limitations of standard convolutional neural networks, in a few recent works it had been also used in NLP [11, 43, 46, 12, 14, 27, 44, 14, 6, 15]. Which is why instead of pure-attention-mechanism-based aggregation, we also explore capsule routing based aggregations for classification.

3 Dataset Details

We prepared our training dataset by aggregating datasets from various sources available through CrisisLex[2] and CrisisNLP[3]. Particularly, we used CrisisLexT6 [30], CrisisLexT26 [31] from CrisisLex, and resource # 1 [22], resource # 2 [20], resource # 3 [21], resource # 4 [29], resource # 5 [3], resource # 7 [1], and resource # 10 [2] from CrisisNLP.

The first issue that we had to face with aggregating all the datasets is combining the classification labels. Different datasets were annotated in different procedures. There are different classes which are almost similar, there are classes in hierarchical relationships, some are more fine-grained, some are more coarse-grained and so on. Ideally, it is necessary to carefully study all the annotation schemes to combine them but for the project, we used a simple label reduction scheme based on our subjective judgments.

[2] https://www.crisislex.org/

[3] https://crisisnlp.qcri.org/

Total number of Tweets in corpus	$1,41,144$
Binary Class Statistics	
Informative	$83,643$
Not Informative	$57,501$
Multi-Class Statistics	
Unlabeled	$53,072$
Casualties and Damage	$17,081$
Caution and Advice	$4,126$
Donation	$9,364$
Not Informative	$57,501$

Table 1: Dataset statistics

We mapped all classes seemingly related to affected individuals, property damage, requests for help, food, and so on to a single class "Casualties and Damage". We mapped all classes seemingly related to disaster warnings, advice, and cautions to a single class "Caution and Advice". We mapped all classes related to volunteering, donation, and support to a single class "Donation". Besides these, we also had a lot of samples that had some label to denote that it is informative and related to a disaster event but were hard to map into one of the above classes or any new class which is mutually exclusive from the above ones. Many of these samples had a class that was too general and broad. We labeled such samples as "Unlabeled". The rest of the classes were annotated as "Not Informative". We also mapped classes related to emotional support and prayers as "Not Informative". We did not use the "Unlabeled" samples for multi-class classification.

Besides using a multi-class classification scheme, we parallelly prepared a more coarse-grained binary classification scheme where "Caution and Advice", "Casualties and Damage", "Donation", and "Unlabeled" classes are all mapped to "Informative", and the rest remains as "Not Informative". As a result of including the "Unlabeled" samples, we end up having far more training samples for Binary classification than that for Multi-class classification. See Table 1 for some dataset details.

The exact mapping label reduction scheme is accessible through our Github repository.

In the pre-processing stage, we removed all URLs, user mentions, and retweet symbols. We also removed any duplicates.

We used $20,000$ samples for testing, $10,000$ samples for validation, and the rest for training.

4 Method

4.1 Multi-Task Learning

One dilemma that we had to initially face is that there were a lot more data samples if we wanted to do binary classification but doing multi-class classification would have given us more fine-grained and informative classes. Instead of giving in to this dilemma, we formulated a multi-task learning approach [7] so that our model can gain supervised signals from both binary classification task and multi-class classification task. Since the tasks are similar enough, we believe that the supervised signal from binary classification can help multi-class classification and vice-versa. We took a rather simple approach to multi-task learning where we simply linearly combined the two task objectives to create the final objective function:

$$\mathcal{L}(\theta, \theta_{binary}, \theta_{multi}) = \alpha \cdot \mathcal{L}_{binary}(\theta, \theta_{binary}) + \beta \cdot \mathcal{L}_{multi}(\theta, \theta_{multi}) \cdot M$$

We use $\mathcal{L}(\theta)$ as a shorthand to mean a loss function with learnable parameters as θ. For $\mathcal{L}_{binary}$ we used binary cros- entropy, and for $\mathcal{L}_{multi}$, we used the weighted cross-entropy loss function. α and β are scalar weights. They are hyperparameters in the interval [0,1]. M is the label mask. It is used to zero-out the loss for "Unlabeled" samples for multi-class classification so that the unlabeled samples do not contribute to the gradient updates for multi-class classification.

We used a shared (by both tasks) BERT-based encoder followed by an MLP or Bi-LSTM to create a shared single-vector encoded representation of any given tweet. θ denotes the parameters involved

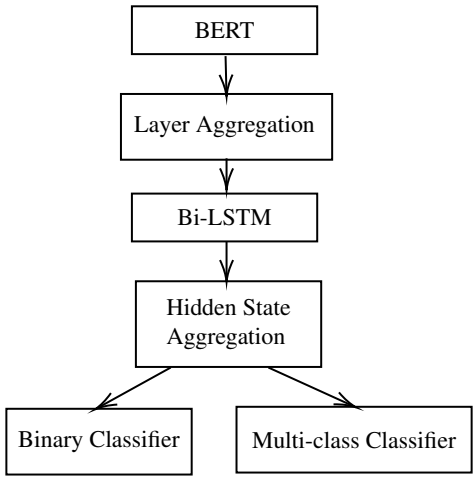

Figure 1: Our overall architecture with all the components

in this process. On top of this final shared representation, we used two different classifiers - a binary classifier and a multi-class classifier. For the binary classifier, we used a linear layer transforming the shared representation into a scalar value followed by a Sigmoid activation function. For the multi-class classifier, we used a linear layer transforming the representation into a un-normalized probability distribution over the classes, which is then normalized with a Softmax activation function.

4.2 BERT

BERT [9] is a Transformer [38] based masked language model which is pre-trained on a large text corpus. The BERT-encoded representations can be used as contextual (sub)-word embeddings. BERT can be also used for transfer learning on other tasks by fine-tuning its parameters.

A Transformer consists of a multiheaded attention (MH) function, which is usually followed by feed-forward layers (with dropouts, layer normalizations and residual connections in between). The MH function uses scaled-dot-product-attention over a given list of queries (Q), keys (K), and values (V). For every token in a query sequence, the attention function computes a weighted summation of the corresponding value sequence. The weights are computed based on the compatibility between the query tokens and the tokens in the corresponding key sequence. Formally, the scaled-dot-product-attention function can be formulated as:

$$Attention(Q, K, V) = \sigma\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where d_k is the dimension of Q and K, and σ is the Softmax function. The multi-headed attention function performs multiple scaled-dot-product attention parallelly over multiple heads - each having different linearly projected representations of Q, K, and V. The results of attention in each head are then concatenated and linearly projected. This can be formalized as:

$$h_i = Attention(QW_i^q, KW_i^k, VW_i^v)$$

$$MH(Q, K, V) = Concat(h_0, ..., h_j)W_o$$

with $i = 0, \cdots, j$ where j is the number of heads. Using the same sequence for Q, K, and V allows self-attention over the given input.
Generally, we used the final layer (or layer-aggregated) hidden state token representations of BERT for the downstream modules (usually a Bi-LSTM), but when we use the baseline without Bi-LSTM (BERT followed by an MLP), we used the pooled output [CLS] token which is specifically setup for classification.

4.3 Bi-LSTM

Given an input sequence, $(x_1, x_2, x_3, ..., x_n)$, a Recurrent Neural Network (RNN) processes each token x_t at every time-step t by producing a hidden state, h_t, using the context of the previous hidden state, h_{t-1}, from the previous time-step $t-1$. Formally, this can be expressed as:

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

where f is a non-linear activation function. The final output of an RNN is a sequence of hidden states, $(h_1, h_2, h_3, ..., h_n)$. Long-Short Term Memory (LSTM) network [18] is a variant of RNN which 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:

$$f_t = \sigma(x_t W_f + h_{t-1} U_f + b_f) \tag{1}$$

$$i_t = \sigma(x_t W_i + h_{t-1} U_i + b_i) \tag{2}$$

$$o_t = \sigma(x_t W_o + h_{t-1} U_o + b_o) \tag{3}$$

$$g_t = \tanh(x_t W_c + h_{t-1} U_c + b_c) \tag{4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

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.

Bidirectional LSTM (Bi-LSTM) [13] has two LSTM 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 (we used the concatenation operator). Formally, we have:

$$\overrightarrow{h_t} = LSTM(x_t, \overrightarrow{h}_{t-1}) \tag{7}$$

$$\overleftarrow{h_t} = LSTM(x_t, \overleftarrow{h}_{t-1}) \tag{8}$$

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}] \tag{9}$$

Where $[;]$ denotes concatenation.

Most of our models use a Bi-LSTM on top of the BERT-encoded hidden state representations of the tokens in a given Tweet. Bi-LSTM can create a deeper representation of by bi-directionally processing the intermediate representation (from BERT). Furthermore, there are works [8] that show that Transformer based models can achieve further gain in performance from utilizing an RNN or LSTM. This further motivated us in combining BERT with a Bi-LSTM. A BERT-BiLSTM stack was also used by Ma et al. [28].

4.4 Layer Aggregation (LA)

Different layers of a deep neural model are often better at different aspects of a task. In NLP, for example, early layers are usually better at syntactic processing whereas the later layers are better for semantic processing [32, 37]. Thus, there are some motivations for aggregating different layer representations. Furthermore, different works [33, 10, 11, 45, 41] empirically showed the benefit from combining different layers. Thus, we also attempted to aggregate the last few layers of BERT instead of just using the last one. For layer aggregation, we tried two different approaches. In one approach, we used a simple attention mechanism for layer aggregation. In this approach, we first dynamically computed scores for each token in each layer as follows:

$$s_i^j = tanh([h_i^1; h_i^2; ....; h_i^n]W_j + B_{1j})V_j + B_{2j}$$

	Binary				Multi			
Models	P	R	F1	Acc	P	R	F1	Acc
BERT-MLP	0.900	0.920	0.909	0.891	0.808	0.827	0.817	0.886
BERT-BiLSTM	0.911	0.896	0.904	0.886	0.803	0.830	0.817	0.887
BERT-BiLSTM+attn HA	0.920	0.885	0.902	0.886	0.803	0.841	0.821	0.888
BERT-BiLSTM+attn LA	0.892	0.927	0.909	0.890	0.784	0.840	0.811	0.880
BERT-BiLSTM+attn LA+attn HA	0.903	0.913	0.908	0.890	0.802	0.835	0.818	0.884
BERT-BiLSTM+caps LA+attn HA	0.896	0.920	0.908	0.889	0.768	0.839	0.802	0.874
BERT-BiLSTM+caps LA+caps HA	0.869	0.945	0.906	0.883	0.738	0.842	0.787	0.858

Table 2: Results (P denotes Precision, R denotes Recall, Acc denotes Accuracy, LA denotes Layer Aggregation, HA denotes Hidden State Aggregation, attn denotes attention, and caps denote capsule routing)

Here, s_i^j denotes the score for the ith token of jth layer. Then we normalized (using Softmax) the scores over all the layers for tokens from each timestep:

$$a_i^j = \frac{exp(s_i^j)}{\sum_j exp(s_i^j)}$$

Finally, we did a weighted summation over the token-representations from the involved layers based on the normalized scores.

$$h_i' = \sum_j a_i^j h_i^j$$

For the second approach, we used capsule routing. In this approach, first, we used a simple linear layer over each layer token-representations. We then treat the linear layer outputs as capsules. Therefore, essentially, we treat linear transformations of token-representations from each layer as capsules. Similarly, we used another linear transformation over the token representations of each layer followed by a Sigmoid to represent input capsule scores which denotes how much of the input capsule is going to be used. After that, we route the input capsules to multiple single-dimensional output capsules (we route the layers in this manner because it was found to perform best in previous works [11]). We concatenate the output capsules to form the final layer aggregated representation where token-representations from different layers are combined into a single vector representing for that token. Instead of dynamic routing, or EM routing as is traditionally used, we used a newer routing algorithm (Heinsen Routing) [15]. Heinsen Routing is still based on EM-routing but it also has an extra 'D-step'. Essentially it is based on the notion that "output capsules should benefit from the input data they use, and lose benefits from any input data they ignore," [15]. The D-step computes how much of the input capsule is going to be used and how much is going to be ignored. The output capsules are then scored based on the net benefit from using the input capsules and the net cost of ignoring the input capsules. The output capsule scores are then used in the next iteration to modulate the routing probability from an input capsule to an output capsule.

4.5 Hidden State Aggregation (HA)

Using just the final backward and forward hidden states of a Bi-LSTM for classification means that only the final hidden state is burdened to encode all the information necessary for classification. However, RNNs often also have a recency bias and may not encode information from distant history as efficiently. But information from the distant states are readily available from stored past hidden states. That is why there are some incentives to aggregate information from all the hidden states instead of just using the final one for classification. Similar to layer aggregation, in this case too, we explored two approaches - one using an attention mechanism, and another using capsule routing in a similar fashion as before. The exact details and hyperparameters can be seen from the code.

5 Implementation

We compared different models all of which use BERT, particularly cased multilingual BERT (base model) so that we can handle multiple languages. We used HuggingFace's library[4] [42] to load

[4] https://github.com/huggingface/transformers

Tweet	Predicted Class	Gold Class
@user (as of 6:20 p.m.) classes in all levels in the entire metro manila are suspended tomorrow, dec. 8, 2014. #walangpasok	Caution and Advice	Unlabeled
@user nepal f india is with you! god is with you! hope all srvivs and gets all d help reqird...all prayers are wid evry 1 der	Not Informative	Unlabeled
@user se registra un fuerte #sismo en #costarica, tras el mismo se declar una alerta de tsunami para #costarica, #panama y ...	Caution and Advice	Caution and Advice
@user #flood #disaster odisha floods kill 45, leave 300000 villagers marooned - reuters india: reuters indiaodisha f... http	Casualties and Damage	Casualties and Damage

Table 3: Some examples

pre-trained BERT for fine-tuning. We used class weights for multi-class classification. The weight for any class i is computed by the following formula:

$$max_j(count(label_j))/count(class_i)$$

We used total hidden units of 256 for the Bi-LSTM. We used AdamW optimizer with a learning rate of $1e^{-3}$ for non-BERT parameters and a fine-tuning learning rate of $2e^{-5}$ for BERT parameters. For multi-task learning, we used α, β, both as 0.5. We used the official code for Heinsen Routing [5] as a reference to implement it for our purpose. To reduce memory usage we only fine-tune and aggregate the last six layers of BERT (the earlier layers are kept frozen). Other details can be found in our repository.

6 Results

We calculated macro-precision, macro-recall, macro-F1, and accuracy for all the models for both binary classification and multi-class classification. We show the results on Table 2.
We observe that the baseline BERT-MLP model outperforms most of the more sophisticated variants. The results do not look significantly different. And the extensions do not seem to be helping too much. The potential reasons for such results are:

- With powerful pre-trained language models - 'minor modifications' do not make much difference.
- The more complex models may work better in more complex tasks where there are fewer heuristics to exploit, and may be with more data.
- There may be significant variance in the results simply due to random initialization (will require multiple runs for testing with standard deviation and mean to know) - the relative differences may not be too meaningful if the variance is high.
- Simpler models with higher bias often outperform more complex models in low-resource (and/or simpler) tasks. For example, simple 1D CNN [24] or Deep Averaging Networks [23] often had been highly competitive with more complex architectures in different classification related NLP tasks.

7 Conclusion

We prepared a Twitter dataset for classification with disaster-related labels by aggregating data from various sources. We compared multiple BERT based models on the dataset. We explored layer aggregation and hidden state aggregation in an effort to improve classification results. We explored attention-based aggregation and capsule routing based aggregation. Although our extensions do not offer much benefit, our general results for both multi-class classification and binary classification are moderately high.

[5] https://github.com/glassroom/heinsen_routing

8 Limitations and Future Works

The label reduction procedure that we used has some room for improvement. Ideally, the label map can be improved after more careful studying of the different annotation schemes used in different papers and thinking more deeply on the best way to combine them. There is room for more exhaustive hyperparameter tuning and experimentation. There are also some missing baselines and ablation. Precisely, it would have been interesting to see the effect of removing the multi-task training and training individual models on multi-class classification and binary classification separately. This could answer questions like these: is multi-tasking really helping both task? Do the extra data with binary class actually help in multi-class classification? In hindsight, it should have been an important ablation study that we now leave for future work. Also, we initially planned to prepare a 'Tweet-BERT' by fine-tuning BERT on Twitter dataset on masked-language modeling task. Although we prepared a demo code for masked language model training, under the constraints of time, we were unable to train the 'Tweet-BERT' and compare it with normal BERT. This is another work that we leave for the future.

References

[1] Firoj Alam, Shafiq Joty, and Muhammad Imran. Domain adaptation with adversarial training and graph embeddings. 2018.

[2] Firoj Alam, Shafiq Joty, and Muhammad Imran. Graph based semi-supervised learning with convolution neural networks to classify crisis related tweets. In *Twelfth International AAAI Conference on Web and Social Media*, 2018.

[3] Firoj Alam, Ferda Ofli, and Muhammad Imran. Crisismmd: Multimodal twitter datasets from natural disasters. In *Proceedings of the 12th International AAAI Conference on Web and Social Media (ICWSM)*, June 2018.

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

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. 2015.

[6] Juncheng Cao, Hai Zhao, and Kai Yu. Cross aggregation of multi-head attention for neural machine translation. In Jie Tang, Min-Yen Kan, Dongyan Zhao, Sujian Li, and Hongying Zan, editors, *Natural Language Processing and Chinese Computing*, pages 380–392, Cham, 2019. Springer International Publishing.

[7] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[8] Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, et al. The best of both worlds: Combining recent advances in neural machine translation. *arXiv preprint arXiv:1804.09849*, 2018.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. Exploiting deep representations for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4253–4262, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[11] Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Longyue Wang, Shuming Shi, and Tong Zhang. Dynamic layer aggregation for neural machine translation with routing-by-agreement. *arXiv preprint arXiv:1902.05770*, 2019.

[12] Jingjing Gong, Xipeng Qiu, Shaojing Wang, and Xuanjing Huang. Information aggregation via dynamic routing for sequence encoding. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2742–2752, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[13] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. In *International Conference on Artificial Neural Networks*, pages 799–804. Springer, 2005.

[14] Shuhao Gu and Yang Feng. Improving multi-head attention with capsule networks. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 314–326. Springer, 2019.

[15] Franz A Heinsen. An algorithm for routing capsules in all domains. *arXiv preprint arXiv:1911.00792*, 2019.

[16] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pages 44–51. Springer, 2011.

[17] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with EM routing. In *International Conference on Learning Representations*, 2018.

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

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

[20] Muhammad Imran, Shady Elbassuoni, Carlos Castillo, Fernando Diaz, and Patrick Meier. Practical extraction of disaster-relevant information from social media. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 1021–1024. International World Wide Web Conferences Steering Committee, 2013.

[21] Muhammad Imran, Shady Mamoon Elbassuoni, Carlos Castillo, Fernando Diaz, and Patrick Meier. Extracting information nuggets from disaster-related messages in social media. *Proc. of ISCRAM, Baden-Baden, Germany*, 2013.

[22] Muhammad Imran, Prasenjit Mitra, and Carlos Castillo. Twitter as a lifeline: Human-annotated twitter corpora for nlp of crisis-related messages. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may 2016. European Language Resources Association (ELRA).

[23] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, Beijing, China, July 2015. Association for Computational Linguistics.

[24] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[25] Shamanth Kumar, Geoffrey Barbier, Mohammad Ali Abbasi, and Huan Liu. Tweettracker: An analysis tool for humanitarian and disaster relief. In *ICWSM*, 2011.

[26] Hongmin Li, Doina Caragea, Cornelia Caragea, and Nic Herndon. Disaster response aided by tweet classification with a domain adaptation approach. *Journal of Contingencies and Crisis Management*, 26(1):16–27, 2018.

[27] Jian Li, Baosong Yang, Zi-Yi Dou, Xing Wang, Michael R Lyu, and Zhaopeng Tu. Information aggregation for multi-head attention with routing-by-agreement. *arXiv preprint arXiv:1904.03100*, 2019.

[28] Guoqin Ma. Tweets classification with bert in the field of disaster management. 2018.

[29] Dat Tien Nguyen, Kamela Ali Al Mannai, Shafiq Joty, Hassan Sajjad, Muhammad Imran, and Prasenjit Mitra. Robust classification of crisis-related data on social networks using convolutional neural networks. In *Eleventh International AAAI Conference on Web and Social Media*, 2017.

[30] Alexandra Olteanu, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. Crisislex: A lexicon for collecting and filtering microblogged communications in crises. In *Eighth International AAAI Conference on Weblogs and Social Media*, 2014.

[31] Alexandra Olteanu, Sarah Vieweg, and Carlos Castillo. What to expect when the unexpected happens: Social media communications across crises. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '15, pages 994–1009, New York, NY, USA, 2015. ACM.

[32] Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

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

[34] Jishnu Ray Chowdhury, Cornelia Caragea, and Doina Caragea. Keyphrase extraction from disaster-related tweets. In *The World Wide Web Conference*, pages 1555–1566. ACM, 2019.

[35] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866, 2017.

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

[37] Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy, July 2019. Association for Computational Linguistics.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[39] Sudha Verma, Sarah Vieweg, William J Corvey, Leysia Palen, James H Martin, Martha Palmer, Aaron Schram, and Kenneth Mark Anderson. Natural language processing to the rescue? extracting" situational awareness" tweets during mass emergency. In *ICWSM*, pages 385–392. Barcelona, 2011.

[40] Sarah Vieweg, Amanda L Hughes, Kate Starbird, and Leysia Palen. Microblogging during two natural hazards events: what twitter may contribute to situational awareness. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1079–1088. ACM, 2010.

[41] Qiang Wang, Fuxue Li, Tong Xiao, Yanyang Li, Yinqiao Li, and Jingbo Zhu. Multi-layer representation fusion for neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3015–3026, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[42] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

[43] Min Yang, Wei Zhao, Jianbo Ye, Zeyang Lei, Zhou Zhao, and Soufei Zhang. Investigating capsule networks with dynamic routing for text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3110–3119, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[44] Zhengxin Yang, Jinchao Zhang, Fandong Meng, Shuhao Gu, Yang Feng, and Jie Zhou. Enhancing context modeling with a query-guided capsule network for document-level translation. *arXiv preprint arXiv:1909.00564*, 2019.

[45] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2403–2412, 2018.

[46] Ningyu Zhang, Shumin Deng, Zhanling Sun, Xi Chen, Wei Zhang, and Huajun Chen. Attention-based capsule networks with dynamic routing for relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 986–992, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

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

[48] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212, Berlin, Germany, August 2016. Association for Computational Linguistics.