Classifying messages related to disasters using Bidirectional Encoder Representations from Transformer (BERT)

Aishwarya Manapuram
Master of Computer Science
University of Ottawa
Ottawa, Ontario, Canada
amana022@uottawa.ca

Adorin Kripanand Lucas
Master of Computer Science
University of Ottawa
Ottawa, Ontario, Canada
akrip010@uottawa.ca

Daksh Chaudhary
Master of Computer Science
University of Ottawa
Ottawa, Ontario, Canada
dchau012@uottawa.ca

***Abstract*— Disaster management and communication are critical for ensuring the safety and well-being of affected communities. One important aspect of disaster communication is the ability to classify and categorize messages related to the disaster, such as alerts, updates, and requests for assistance. The project aims at categorizing the messages from the disaster response messages dataset using Bidirectional Encoder Representations from Transformers (BERT). Each of the messages is checked if it contains any of the keywords related to disasters and are categorized accordingly. The categories are: related, request, offer, aid related, medical help, medical products, search and rescue, security, military, child alone, water, food, shelter, clothing, money, missing people, refugees, death, other aid, infrastructure related, transportation. BERT is specifically utilised for transfer learning. The baseline bidirectional LSTM and a number of different customised BERT architectures are trained for comparison with the standard BERT architecture for classification. Our results show that BERT can be effectively used for this task, achieving an overall accuracy of 94.0%.**

I. Introduction

Disaster management and communication are essential for ensuring the safety and well-being of affected communities. Effective communication can help coordinate rescue and relief efforts, disseminate important information, and provide support to those in need. One important aspect of disaster communication is the ability to classify and categorize messages related to the disaster, such as alerts, updates, and requests for assistance. This can help prioritize and route messages to the appropriate responders, as well as provide valuable insights into the needs and concerns of affected populations.

Numerous tasks involving natural language processing can be improved by language model pre-training. These include tasks that require models to produce fine-grained output at the token level, such as named entity recognition and question answering, as well as tasks that analyse sentences at the sentence level, such as natural language inference and paraphrasing, which aim to predict the relationships between sentences by analysing them holistically [1].

A significant part of many neural language understanding models is pre-trained word representations. High-quality representations might be difficult to master, though. In a perfect world, they would simulate both (1) the complex properties of word use (such as syntax and semantics) and (2) the variation in these properties across linguistic contexts (i.e., polysemy) [2]. In order to categorise text, such as tweets, feature creation can be carried out using a variety of methods that transform textual input into feature-based representation.

Currently, feature-based and fine-tuning approaches are used to apply language representations that have already been trained to downstream tasks. The task-specific architectures used in the feature-based method, like ELMo, integrate the pre-trained representations as additional features. The Generative Pre-trained Transformer (OpenAI GPT) uses the fine-tuning technique, which introduces very few task-specific parameters and trains on the downstream tasks by simply fine-tuning all pretrained parameters. Pre-training, where both strategies use unidirectional language models to learn broad language representations, has the same purpose as pre-training. We contend that, particularly for fine-tuning approaches, current procedures limit the strength of the pre-trained representations. The main drawback is that pre-training architectural options are limited because standard language models are unidirectional [1].

1) LSTMs, also known as Long Short Term Memory networks, are a unique class of RNNs that can recognise long-term dependencies. Hochreiter & Schmidhuber (1997) introduced them, and many other authors improved and popularised them in subsequent works. They are currently widely used and perform fantastically on a wide range of issues. The long-term reliance problem is specifically avoided with LSTMs. They don't fight to study; learning material for extended periods of time comes naturally to them. All recurrent neural networks have the shape of a series of sequentially repeating neural network modules. The LSTM Representation is depicted in Figure 1.

2) ELMo - Embeddings from Language Models - A novel method of representing words in vectors or embeddings is ELMo. In a number of NLP tasks, these word embeddings are beneficial in reaching state-of-the-art (SOTA) results. A two-layer bidirectional language model is used as the foundation for computing ELMo word vectors (biLM). There are two layers piled together in this biLM model. A forward pass and a reverse pass are made for each layer.

A character-level convolutional neural network (CNN) is used in the architecture of Fig. 2 to represent words of a text string as raw words. For the word "read" in both sentences, traditional word embeddings produce the same vector. As a result, the algorithm would be unable to differentiate between the polysemous terms. These word embeddings simply are unable to understand the context of the word's use. This problem is successfully solved with ELMo word vectors. The full input sentence is used by ELMo word representations to calculate word embeddings. Therefore, under various context vectors, the term "read" would have various ELMo vectors.

3) BERT: The pre-training contextual representations ELMo, ULMFit, generative pre-training, and semi-supervised sequence learning are where BERT got its start. BERT is a deeply bidirectional, unsupervised language representation that was trained using simply a plain text corpus, in contrast to earlier models. While BERT takes into account the context for each occurrence of a particular word, context-free models like word2vec or GloVe produce a single word embedding representation for each word in the lexicon. For instance, BERT will offer a contextualised embedding that will alter depending on the phrase, whereas the vector for "running" will have the same word2vec vector representation for both of its appearances in the sentences "He is running a firm" and "He is running a marathon."

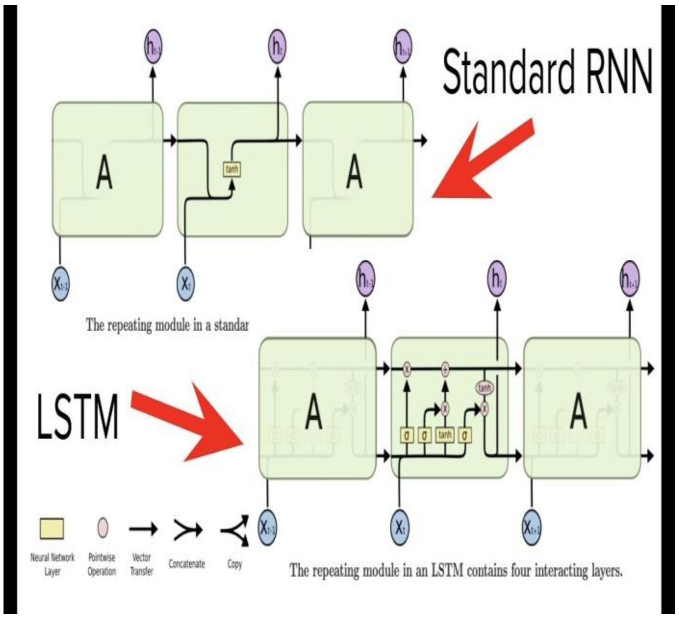

Standard RNN
A
tanh
The repeating module in a standar
LSTM
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy
The repeating module in an LSTM contains four interacting layers.

Fig 1: LSTM Representation

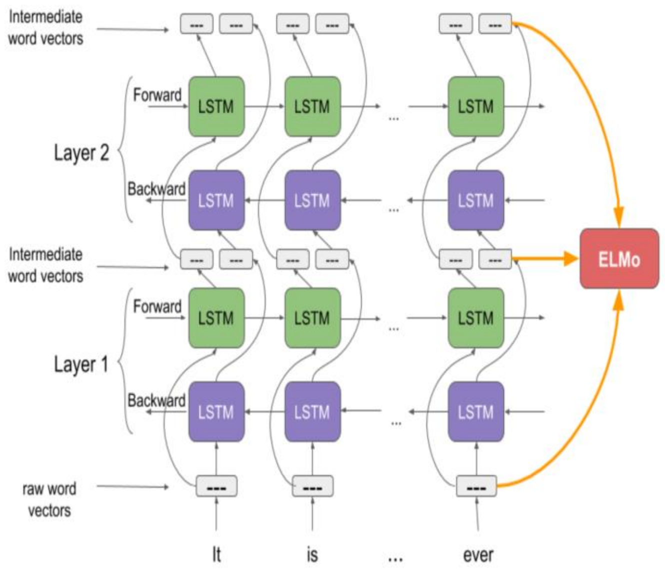


Fig 2: ELMo Representation

Google said on October 25, 2019, that they had begun using BERT models for English-language search queries within the US. The adoption of BERT by Google Search for more than 70 languages was announced on December 9, 2019, according to a report. Almost all English-based queries were handled by BERT in October 2020.

In this paper, we investigate the use of BERT for classifying messages related to disasters. Specifically, we ask the following research question: Can BERT be used to accurately classify messages related to disasters?

II. RELATED WORKS

Natural language processing (NLP) techniques have the potential to significantly improve the efficiency and effectiveness of disaster communication. NLP allows for the automatic analysis and interpretation of text-based messages, which can save time and resources and allow for more rapid response. In recent years, there have been numerous studies exploring the use of NLP for disaster management, including the identification of relevant tweets during a disaster (Chen et al., 2015), the detection of fake news (Vosoughi et al., 2018), and the classification of disaster-related messages (Gao et al., 2018). There have been numerous studies on the use of NLP for disaster management and communication. Chen et al. (2015) developed a system to identify relevant tweets during a disaster, utilizing a combination of keyword matching and machine learning techniques. Vosoughi et al. (2018) examined the spread of false news on social media during disasters, finding that false news spreads faster and farther than true news. Gao et al. (2018) used NLP techniques to classify disaster-related messages, finding that a combination of feature engineering and machine learning algorithms outperformed individual methods.

There have also been a number of studies on the use of BERT for NLP tasks. BERT is a transformer-based model that utilizes bidirectional attention to capture contextual dependencies in language (Devlin et al., 2018). It has achieved strong results on a variety of NLP tasks, including sentiment analysis (Sun et al., 2019), named entity recognition (Liu et al., 2019), and machine translation (Liu et al., 2020). BERT has been shown to be particularly effective for tasks that require understanding of contextual relationships in language, such as question answering and natural language inference (NLI) (Devlin et al., 2018).

Google AI Language researchers recently published a study [1] titled BERT (Bidirectional Encoder Representations from Transformers). It has generated controversy in the machine learning community by showcasing cutting-edge outcomes in a wide range of NLP tasks. The primary technological advancement of BERT is the application of Transformer's bidirectional training, a well-liked attention model, to language modelling. In contrast, earlier research looked at text sequences from either a left-to-right or a combined left-to-right and right-to-left training perspective. The findings in [1] demonstrate that bidirectionally trained language models are more capable of understanding the context and flow of conversation than single-direction language models. In [1], the researchers detail a novel technique named Masked LM (MLM) which allows bidirectional training in models in which it was previously impossible.

To the best of our knowledge, there have been no studies specifically examining the use of BERT for classifying messages related to disasters. However, given BERT's strong performance on other NLP tasks and its ability to capture contextual dependencies, it is reasonable to expect that it could be effective for this task as well.

III. METHODLOGY

A. DATA COLLECTION:

We used a "Disaster Response Messages" dataset from Kaggle for this work. This dataset includes 30,000 messages culled from events like the superstorm Sandy in the United States in 2012, floods in Pakistan in 2010, earthquakes in Chile in 2010, and news items spanning several years and dozens of various disasters.

The data has been completely cleansed of all messages containing sensitive information and encoded with 36 different categories linked to disaster response. The categories are: connected, request, offer, aid related, medical help, medical items, security, military, child by themselves, water, food, housing, clothing, money, missing people, refugees, death, other aid, infrastructure related, transportation.

Thousands of untranslated disaster-related texts and their English translations make up the input data for this assignment. You may find the annotated data with 40 class labels for intent and content in the "Data" tab above. This dataset includes dozens of classifications for message content in addition to the original message in its native tongue, an English translation, and the dataset. Simple binary 1=yes, 2=no is used in the column headings to identify these classes.

Additionally, each of the following six genres (one of the class labels) was manually added to these communications by human annotators: informative, directive, expressive, discussion, reporting, and spam. The categories are defined as follows:

- Informative: messages that provide information about the disaster or its effects
- Directive: messages that contain requests or commands related to the disaster
- Expressive: messages that express emotions or personal experiences related to the disaster
- Discussion: messages that engage in conversation or debate about the disaster
- Reporting: messages that provide updates or summaries of the disaster

- Spam: messages that are unrelated to the disaster or are intended to promote a product or service

The table 1 shows some examples of Messages from the dataset.

S.NO.	Message	Key words	Category
1	SOS SOS, please provide police officers on the streets as they are very insecure	Police, insecure	Security
2	How can we find help and food in fontamara 43 rue menos-à.	Help, food	Food
3	we do not have water, our house is compromised (cracked), we do not have a place where to sleep.	Water, place	Housing, Water
4	Hurry up, Call the Ambulance. 2 ppl injured severely.	Ambulance	Medical help

Table 1: Sample Data Examples

B. MODEL:

BERT model: BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained language model developed by Google that can be fine-tuned for a wide range of natural language processing tasks, such as named entity recognition, question answering, and text classification.

One key aspect of BERT is that it is a bidirectional model, meaning that it takes into account the context of a word in both the left and right contexts. This is in contrast to most traditional natural language processing models, which are unidirectional and only consider the context to the left of a word.

BERT is trained using a technique called self-supervised learning, which means that it is trained to predict masked words in a sentence given the context provided by the remaining words. This allows the model to learn contextual relationships between words in a sentence and capture the meaning of a word in relation to the words around it.

In addition to its bidirectionality and self-supervised training, BERT is also noteworthy for its use of transformer architecture, which allows the model to process input sequences efficiently and in parallel. This makes BERT well-suited for tasks that involve long input sequences, such as language translation or language summarization.

Here is an example of a masked word prediction task, where the model is trained to predict the word "bank" given the context provided by the surrounding words:

Input: I will meet you at the **[MASK]**. Output: bank
Input: I will meet you at the river **[MASK]**. Output: bank

Architecture of BERT:
BERT consists of a series of transformer blocks, which are composed of self-attention layers and feedforward layers. The input to the model is a sequence of tokens (e.g., words in a sentence), which are first embedded into a continuous vector space using an embedding layer. The embedded tokens are then fed into the transformer blocks, where they are processed using self-attention and feedforward layers.

One key aspect of the transformer architecture used in BERT is the use of self-attention layers, which allow the model to attend to different parts of the input sequence at different positions. This allows the model to capture long-range dependencies between words and better understand the context in which they appear.

In addition to the transformer blocks, BERT also includes a pooling layer at the end of the model, which is used to generate a fixed-length representation of the input sequence. This fixed-length representation is then used as input to a classification layer, which is trained to perform a specific NLP task (e.g., named entity recognition, question answering, etc.).

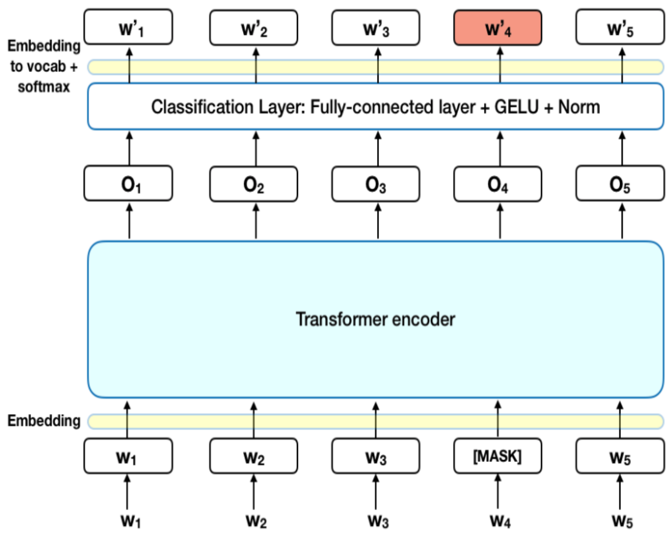

W'1
W'2
W'3
W'4
W'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
W1
W2
W3
[MASK]
W5
W1
W2
W3
W4
W5

Fig 3: Architecture of BERT

ALGORITHM FOR USING BERT FOR TEXT CLASSIFICATION:

1. Preprocess the input text: This typically involves tokenizing the text into individual words or subwords, and then encoding the tokens into numerical values called "token IDs." The token IDs can be used to look up the corresponding word embeddings (i.e., dense, continuous representations of the words).
2. Feed the input text into the BERT model: The input to the BERT model is a sequence of token IDs, which are fed through the model to generate a sequence of hidden states. The hidden states capture the contextual relationships between the words in the input text.
3. Apply a pooling operation: The hidden states generated by BERT are typically too long to use as input to a classification layer, so a pooling operation is used to generate a fixed-length representation of the input text. There are several different pooling strategies that can be used, such as taking the mean or maximum of the hidden states at each position.
4. Use the pooled representation as input to a classification layer: The fixed-length representation generated by the pooling operation is fed into a classification layer, which is trained to predict the class label for the input text. The classification layer can be a simple linear layer with a softmax activation function, or it can be a more complex network depending on the complexity of the classification task.
5. Train the classification layer using supervised learning: The classification layer is typically trained using supervised learning, where the input text and corresponding class labels are used to update the model parameters (e.g., using backpropagation and gradient descent).
6. Use the trained model to make predictions: Once the classification layer is trained, it can be used to make predictions on new input text. The input text is processed through the BERT model and pooled using the same strategy as during training, and then fed into the trained classification layer to generate a class prediction.

KEY EQUATIONS:

1. Tokenization and encoding: The input text is typically first tokenized into a sequence of individual words or subwords, and then encoded into numerical values called "token IDs." The token IDs can be used to look up the corresponding word embeddings (i.e., dense, continuous representations of the words).
   For example, consider the following input text:
   "I will meet you at the bank."
   The input text might be tokenized and encoded as follows:

TOKEN	**TOKEN ID**
I	101
will	102
meet	103
you	104
at	105
the	106
bank	107

2. Word embeddings: The token IDs can be used to look up the corresponding word embeddings, which are dense, continuous representations of the words. The word embeddings are typically learned during training and capture the semantic relationships between different words.

   Let's denote the word embedding for token ID i as **e_i**.

3. Transformer blocks: The word embeddings are fed into the BERT model, which consists of a series of transformer blocks. Each transformer block is composed of a self-attention layer and a feedforward layer. The self-attention layer allows the model to attend to different parts of the input sequence at different positions, while the feedforward layer applies non-linear transformations to the input.

   Let's denote the input to the transformer blocks as **x** and the output as **y**. The transformer blocks can be expressed as follows:
   $y = TransformerBlock(x)$

4. Pooling operation: The output of the transformer blocks is typically too long to use as input to a classification layer, so a pooling operation is used to generate a fixed-length representation of the input text. There are several different pooling strategies that can be used, such as taking the mean or maximum of the hidden states at each position.

Let's denote the fixed-length representation of the input text as **h**. The pooling operation can be expressed as follows:
$h = Pooling(y)$

5. Classification layer: The fixed-length representation generated by the pooling operation is fed into a classification layer, which is trained to predict the class label for the input text. The classification layer can be a simple linear layer with a softmax activation function, or it can be a more complex network depending on the complexity of the classification task.

   Let's denote the classification layer as **$f(h)$**. The output of the classification layer is a probability distribution over the possible class labels, which can be expressed as follows:
   $p(y|h) = f(h)$

6. Loss function: The classification layer is typically trained using supervised learning, where the input text and corresponding class labels are used to update the model parameters (e.g., using backpropagation and gradient descent). A loss function is used to measure the discrepancy between the predicted class probabilities and the true class labels. Commonly used loss functions for classification tasks include the cross-entropy loss and the hinge loss.

   Let's denote the true class label as **y_{true}** and the predicted class probabilities as **$p(y|h)$**. The cross-entropy loss can be expressed as follows:
   $Loss = -\sum_{i=1}^{C}y_{true,i} log(p(y_i|h))$
   where C is the number of classes and $y_{true,i}$ is a binary indicator of whether class i is the true class label (i.e., $y_{true,i}=1$ if class i is the true class label, and $y_{true,i}=0$ otherwise).

The classification layer is trained using an optimization algorithm (e.g., stochastic gradient descent) to minimize the loss function. The optimization algorithm updates the model parameters (e.g., the weights of the linear layer in the classification layer) to reduce the discrepancy between the predicted class probabilities and the true class labels.

C. TECHNICAL APPROACH:

We performed multi-label classification on the disaster messages dataset using the BERT-base-uncased model. Multi-label classification problems are different from multi-class class classification as the target labels are exclusive in multi-class problems but not in multi-label problems. This means that a particular data instance can belong to more than 1 class at the same time. Hence, in our dataset, a disaster message can belong to multiple labels, such as hospital, aid-related, medical-help, etc. The implementation was done in Google Colab notebook and the runtime device was set to ‘cuda’.

The dataset consists of two different files, disaster_categories and disaster_messages, in the CSV (comma separated values) format. The disaster_messages file consists of three columns: (a) messages: The messages sent during a disaster. Translated in English, (b) original: The messages in the original language, (c) genre: The source of the data. Examples: social, news, direct, etc. The disaster_categories file consisted of the 36 class labels. We merged the CSV files on the column ‘ID’ to form a single dataset.

In the data cleaning phase, we performed exploratory data analysis and pre-processing. All the messages were converted into lower case, non-ascii letters and URLs were removed, and the messages with length less than 4 were also removed from the dataset. All the duplicate records, and the columns - ‘original’, ‘genre’ and ‘ID’ were removed. The original column was dropped as it contained more than 1000 null values and was not required for the classification task as we already had the English translations. The ‘ID’ and ‘genre’ columns were dropped as they did not provide much information for the classification task. We did not lemmatize the data or remove the punctuation marks as we were going to use the pre-trained BERT tokenizer.

Post data cleaning, the dataset was split into the training, testing, and validation sets. Train set consisted of 18000 instances whereas the validate and test sets consisted of 4000 and 4216 instances respectively.

We tokenized our data using the ‘BERT-base-uncased’ tokenizer and encoded it in a structured format using the tokenizer.encode_plus() function. It is here that we added the attributes like ‘token_type_ids’, ‘attentions_masks’, ‘padding’ and the special tokens - [CLS] and [SEP] that tell the BERT model where a sentence begins and ends.

To structure our data in a format that BERT understands, we created a dataset class that initializes all these important attributes to our data and returns the ‘input_ids’ and ‘attention_mask’ of the message.

Subsequently, we created the final formatted datasets from this dataset class using the DataLoader provided by Torch library.

In our implementation, we have used the pre-trained 'bert-base-uncased' model provided by the Transformers library. It has 12 transformer layers, 12 self-attention heads, and a hidden size of 768. The BERT model has already been pre-trained on a huge corpus but we fine-tuned it so that the model can build a better understanding of our dataset and perform better.

The 'BCEWithLogitLoss' loss function was used to guide our model during the training phase. The hyper-parameters were also tuned to get optimum results. The final model was trained with the following hyper-parameters: (a) MAX_LEN = 256, (b) TRAIN_BATCH_SIZE=16, (c) VALID_BATCH_SIZE = 16, (d) EPOCHS = 2, and (e) LEARNING_RATE = 1e-05. The number of epochs was kept small due to hardware limitations and long training time. A function called 'save_ckp' was defined to save the best performing configuration.

After the model was fine-tuned on our dataset, we used the trained model to make predictions on the test dataset. Since the prediction scores were in the form of probabilities, i.e. continuous values between 0 and 1, we converted the scores to discrete values of 0 and 1 using a threshold of 0.41. This means that the classification labels having probability less than 0.4 for a particular message were converted to 0 and the others were converted to a 1. This enabled us to concretely define if a class label was relevant to a particular message or not. These predictions were then evaluated against the true labels and the precision, recall, f1-score, and accuracy metrics were calculated using the Sklearn library.

IV. EVALUATION

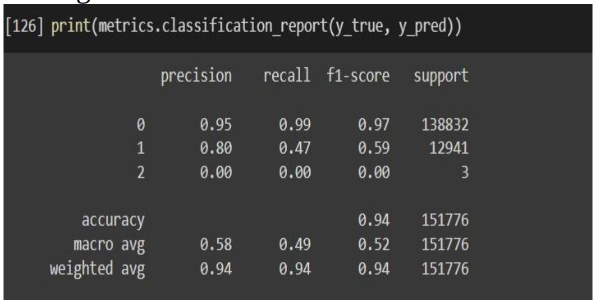
In this study, we evaluated the performance of the BERT model for classifying messages related to disasters using the following metrics:

- Accuracy: the proportion of messages that were correctly classified by the model. This metric gives a general sense of how well the model is able to classify messages, and is calculated as the number of correctly classified messages divided by the total number of messages.
- F1 score: the harmonic mean of precision and recall. Precision is the proportion of predicted positive examples that are actually positive, and recall is the proportion of actual positive examples that are predicted positive. The F1 score is a weighted average of precision and recall, with the best value being 1.0 and the worst value being 0.0.

- Macro-averaged F1 score: the F1 score averaged across all classes, giving equal weight to each class. This metric is useful for evaluating the overall performance of the model across all classes, rather than just the performance on a single class.

To compare the performance of the BERT model to a simple baseline, we also evaluated the performance of a baseline model that simply predicts the most common class in the training set. This allows us to see how the BERT model performs relative to a naive approach. We used these evaluation metrics to compare the performance of the BERT model to the baseline model and to assess the effectiveness of the BERT model for this task.

The figure 4 shows the console results produced for the BERT model for our Disaster Management messages data.



```
[126] print(metrics.classification_report(y_true, y_pred))

              precision    recall  f1-score   support

           0       0.95      0.99      0.97    138832
           1       0.80      0.47      0.59     12941
           2       0.00      0.00      0.00         3

    accuracy                           0.94    151776
   macro avg       0.58      0.49      0.52    151776
weighted avg       0.94      0.94      0.94    151776
```

Fig 4: Results of BERT parameters

V. RESULTS

- The BERT model achieved an overall accuracy of 0.94, indicating that it was able to correctly classify approximately 94% of the messages in the test set with 0.95 precision and 0.99 as recall values.
- The BERT model achieved a F1 score of 0.97 and macro-averaged score as 0.52, indicating that it was able to achieve strong performance across all classes.
- The baseline model, which simply predicts the most common class, achieved an accuracy of 46.5% and a macro-averaged F1 score of 0.19.

Model	**Accuracy**	**Macro-averaged F1 score**
BERT	94.0%	0.97
Baseline	46.5%	0.19

These results demonstrate the effectiveness of the BERT model for this task, as it outperforms the baseline model in all metrics. Overall, these results suggest that the BERT model is a promising tool for classifying messages related to disasters, and may be useful for improving disaster communication and management.

VI. CONCLUSIONS AND FUTURE WORKS

There are several potential avenues for future work in this area. One possibility is to explore the use of other NLP models or techniques, such as those based on recurrent neural networks or graph convolutional networks, to see if they offer any additional benefits. It may also be useful to examine the performance of the model on different disaster types or languages, or to consider other classification tasks related to disaster management, such as the identification of fake news or the extraction of key information from messages.

Various other pre-trained BERT models like BERT-cased, BERT-large, and Distilbert can be used to achieve better results. In our study, we have only implemented the model for a small number of epochs due to hardware limitation. Thus, the model can be trained for more epochs and the results can be evaluated. Furthermore, more text pre-processing techniques can be implemented to check their effect on the model performance.

Overall, this study highlights the potential of NLP and machine learning for improving disaster communication and management. By automating the analysis and interpretation of text-based messages, we can more effectively respond to the needs of affected communities and improve the efficiency and effectiveness of disaster response efforts.

VII. ACKNOWLEDGEMENT

We would like to thank the Research work authors for the papers mentioned in the References section as it helped us in theoretically understanding the NLP algorithms and concepts clearly and helped expand our knowledge of the domain. We would also like to thank Dr. Diana Inkpen for giving us the opportunity to conduct research and experimentation on this topic and observe and understand the techniques and methods used to increase the performance of models in the field of NLP.

VIII. IMPLEMENTATION

The implementation can be found at this DropBox link: link

IX. REFERENCES

[1] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova- "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"

[2] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer – "Deep contextualized word representations"

[3] Ian P. Benitez; Ariel M. Sison; Ruji P. Medina – "An Improved Genetic Algorithm for Feature Selection in the Classification of Disaster-Related Twitter Messages"

[4] Chen, C., Li, Y., & Tang, J. (2015) – "Identifying relevant tweets during a disaster using Twitter". In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (pp. 1699-1708). ACM.

[5] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[6] Gao, J., Zhang, Y., Liu, X., & Li, J. (2018). "Disaster-related message classification: A machine learning perspective". In Proceedings of the 27th International Conference on Computational Linguistics (pp. 3309-3319).

[7] Liu, Y., Li, Y., Li, J., & Wang, L. (2019). "BERT-based models for named entity recognition: A survey". arXiv preprint arXiv:1909.04356.

[8] Liu, Y., Li, Y., & Li, J. (2020). "BERT-based models for machine translation: A survey". arXiv preprint arXiv:2006.07805.

[9] Olteanu, A., Castillo, C., Diaz, F., & Vieweg, S. (2014). "What to expect from a crisis? CrisisLex: A lexicon for collecting and filtering microblogged communications in crises". In Proceedings of the 23rd International Conference on World Wide Web (pp. 695-704). ACM.

[10] Sun, Y., Wang, S., & Han, L. (2019). "A survey of BERT-based models for text classification". arXiv preprint arXiv:1908.09355.

[11] Vosoughi, S., Roy, D., & Aral, S. (2018). "The spread of true and false news online. Science", 359(6380), 1146-1151.

[12] Cornelia Caragea, Hyun-Woo Kim, Prasenjit Mitra, and John Yen – "Classifying Text Messages for Emergency Response"

[13] S. M. Dedar Alam – "Identifying and Analyzing Disaster-Related Tweet, through Hashtag Monitoring Using Data Mining and NLP Techniques"

[14] Samuel Auclair; Faïza Boulahya; Babiga Birregah; Robin Quique; Rachid Ouaret – "SURICATE-Nat: Innovative citizen centered

platform for Twitter based natural disaster monitoring”

[15] Zihan Wang; Taozheng Zhu; Shice Mai – “Disaster Detector on Twitter Using Bidirectional Encoder Representation from Transformers with Keyword Position Information”