Group 3: Estella Dentinger, Nikita Singh, Andrew Hu, Brittany Nguyen
15.S04 – HODL: Final Project

Categorizing Tweets about Emergencies with Deep Learning

Problem Description

Business Case

The goal of our project is to use different Natural Language Processing methods to immediately identify whether an incoming new tweet indicates a disaster or emergency. In today's social media climate flooded with fake news and disinformation, platforms such as Twitter often mislead the public. However, they are simultaneously one of the most used mediums to disseminate timely information, which could help alert first responders and surrounding communities.

Approximately 35+ million tweets are sent in the U.S. every hour [Krikorian], and billions of dollars could be saved each per year in the U.S. for fire emergencies alone, if average emergency response times are reduced by just a few minutes. Empirical studies have shown that $5,000 to $10,000 worth of property damage due to fire could be avoided, for each minute of reduced response time [Purvis]. In addition, faster response would save thousands of priceless lives, not just from fires, but other emergencies as well.

Technical Challenge

One of the primary challenges of categorizing tweets, is that changes in individual word meanings combined with their positions, could dramatically alter the intended message the entire tweet is trying to communicate. A good example of the technical challenge categorizing similar tweets would be comparing: "The best movie is Ablaze!" and "The theater showing the movie is ablaze!" Traditionally, these sentences might be assigned similar independent variables, because the term "ablaze" or n-gram "movie is ablaze" both trigger an emergency response, their position in the tweet are similar, and other words in the tweet have similar meaning, such as "movie" and "theater". However, these tweets clearly have very different intentions to the average person.

Another major issue with sorting tweets, is their difference versus normal written prose or paragraphs. Posts often include abbreviated phrases, misspelled words, or out of context sentences, which can take on multiple meanings that could even confuse the average person. Moreover, the character limit of tweets forces messages to be short, and allows little room for additional outside context to help with the prediction. We want to capture as many emergencies as possible, but also do not want to have too many false positives. This might overwhelm first responders and make the model impractical. At the same time, we do not wish for our model to underestimate the number of emergencies, which could put lives and property at risk.

Data Description

Our team obtained our dataset from Kaggle, which contains ~10,000 sample tweets with common emergency phrases containing a blend of both true and false positives [Howard et al]. The samples include tweeted text, manually identified keywords associated with emergency in the tweet, location tweet was sent from (if available), and a binary target variable which indicates "true emergency or not." See example of raw dataset in figure below.

```
train_df.head()
```

	id	keyword	location	text	target
6234	8902	snowstorm	South, USA	Sassy city girl country hunk stranded in Smoky...	1
326	472	armageddon	Worldwide	God's Kingdom (Heavenly Gov't) will rule over ...	0
997	1448	body%20bagging	Cloud 9	Mopheme and Bigstar Johnson are a problem in t...	0
7269	10407	whirlwind	Sheff/Bangor/Salamanca/Madrid	@VixMeldrew sounds like a whirlwind life!	0
2189	3137	debris	Nigeria	Malaysia confirms plane debris washed up on Re...	1

The dataset is part of a Kaggle competition, so the official test set that was provided did not have a publicly available binary target output. In order to obtain a test set for our project, we split the original training set into new randomly sampled training, validation, and test sets. We did not use the location variable for this exercise because we do have the expertise to model this information, but it would be an important area for future research because it could add another layer of improved predictability.

No Information Rate / Baseline Model

The distribution of the target variable was examined, in order to establish a No Information Rate or baseline model, upon which to compare subsequent models' performance. It was observed that the target variable was fairly evenly split, with 57% of the tweets marked as 0 (non-emergencies) and 42% marked as 1 (true emergencies), thus a naive classification rule would be to predict every tweet as a non-emergency, which could yield a 0.57 accuracy rate. We made sure our random splits maintained this 0.57 ratio, and avoided both up and downsampling, because information would have been lost on either side (we want both high Precision and Recall). While an 64-16-20 train-valid-test split on ~10,000 samples might seem large, it is actually quite small in the context of Natural Language Processing tasks, especially within the context of meaning, position, and abbreviation/misspelling/limited character size challenges that were described above. In a real world scenario, we would want to add additional True Negatives into the sample, because real emergency tweets are likely rare, so a huge part of the task is reducing False Positives.

Technical Approaches with Deep Learning

Bag-of-Words

The first model employed was a simple bag-of-words model, tweaked by varying its hyperparameters (n-grams, epochs, etc). We preprocessed the data using the STI method from lecture (standardize, tokenize, index) with unigrams, ReLU activation functions, and trained our Neural Network with a final sigmoid activation layer (binary classification). We also experimented with bigrams + keyword vector dimension as the input tensor. After adapting the vectorization layer to the corpus and vectorizing the input, we fed our text into the model, which has eight hidden units (16 was also experimented with), one dense layer, and a sigmoid final activation layer with a total of 40,026 trainable parameters. For compilation, we used the "adam" optimizer with binary cross entropy, experimented with 10 to 20 epochs with a batch size of 32.

GloVe Embedding

Next, we added GloVe embedding to preprocessing, turning the STI method to STIE. GloVe is an unsupervised vector embedding that incorporates global word co-occurrence frequency in general text, into our dataset's vectorized inputs. This helps a neural network recognize some generic aspects of language structure (meaning, occurrence in the same sentence). The obvious flaw here for tweet recognition is that it does not capture any of the word order, abbreviation/misspelling, or other contextual information. For our model, a GloVE embedding layer is added after the original STI process, and the new tensor is fed through a neural network model, consisting of a global average pooling layer, a dense layer with dropout regularization, and a final softmax output layer. The model is compiled with Adam optimizer and categorical cross-entropy loss function and is trained on the training data with a batch size of 32 and for 10 epochs.

HODL Transformer

Another one of our approaches was to utilize the power of generic transformers, which we had spent a significant amount of class time on. The main three aspects we hoped to benefit from with the use of the transformer were: positional encoding, context dependency, and equally lengthed inputs and outputs. Transformers make use of self-attention mechanisms in order to produce contextualized embeddings from stand-alone embeddings, therefore incorporating context into classification tasks.

We utilized the "HODL Transformer" package from class as a starting point. The max_text_length of the tweet input was specified as 140, and a text vectorization layer was established for standardization and tokenization. The layer was adapted to the training corpus, and both the training and validation corpuses were vectorized. An input layer was specified with shape 140. The positional embedding layer took as input the max_text_length of 140, vocabulary size, and embedding dimension of 512. The transformer encoder took as input the 512 embedding dimension, dense layer dimension of 64, and 5 self attention heads. In order to prevent errors in the code and adapt the data from slot classification

to binary classification, flattening was carried out. Four dense layers with ReLU activation functions were produced, alongside a dropout layer with a 0.2 proportion. In the first hidden layer, 3,840 hidden units were used. In the second, 960 were used. In the third and fourth, 240 and 60 were used respectively. Finally, the output layer was a softmax with 2 categories (equivalent to binary classification with a sigmoid). The adam optimizer was also used, alongside categorical cross entropy due to the one-hot encoding of the dependent variable. A batch size of 128 was used and ran for 10 epochs.

Transfer Learning with BERT

Continuing on, we used transfer learning on a pre-trained language model (BERT) to see if it would do a better job out of sample. It was hypothesized that BERT would yield better results since it is trained on entire books/paragraphs/websites, which adds context information to the text vectors. Moreover, transfer learning is powerful on even small datasets because it combines sample-specific knowledge.

First, the necessary packages to install for NLP tasks in tensorflow were loaded. Then, two functions (bert_textvect and bert_features) were specified. Bert_textvect() functions by taking in a text corpus as input and converting it to vector encodings. The text is first tokenized and packed, before the vector encodings are produced. The bert_features() function takes as input a dictionary of the input word ids, input type ids, and input mask. The output features which result serve as the input for the neural network built on top of the pre-trained BERT model. The bert_textvect() function was applied to the text corpus of the training, validation, and test data. The results were then fed into the bert_features() function to produce input features for the neural network.The neural network was then specified with 64 hidden units, one input layer, two dense layers, one dropout layer with 0.1 indicated as the proportion, and a sigmoid output layer (binary classification). Binary cross entropy was employed along with the Adam optimizer and accuracy metric. A batch size of 32 was used for 20 epochs. We also experimented with using three dense layers with 64 hidden units each, but that this resulted in only a very similar (and slightly lower) validation accuracy.

Stacked Bag-of-Words and BERT

Since Bag-of-Words and BERT performed well individually for our problem, we wanted to see if stacking them and creating an ensemble model would yield better results. Traditional ensemble models use an average or majority vote of output probability to determine a final output probability. However, since neural networks are so powerful, and the information captured in both methods could be concatenated then processed through dense layers, we decided to experiment with this approach.

The stacked model code creates a deep neural network architecture consisting of two sub-models, one using Bag-of-Words (BOW) and the other using pre-trained BERT embeddings, and combines them

using a concatenation layer followed by multiple dense layers and a final output layer. The model is trained to classify natural disaster tweets as either real or fake using binary cross-entropy loss and Adam optimizer. The model's training data is fed to the two sub-models simultaneously and is trained for 10 epochs with a batch size of 32.

Results

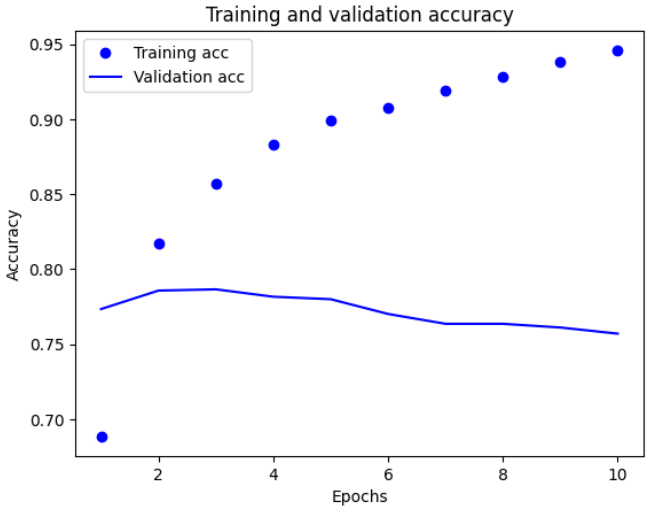
Bag-of-Words Results

The results overall matched our expectations. Ranking of the models from worst to the best performance on the test set were: HODL transformer, Stacked Model, GloVe, Bag-of-Words, and BERT Transfer Learning. This makes logical sense, as the Bag-of-Words model with multi-hot encoding simply takes into account what tokens are present between the incoming input data and the vocabulary. However, no context or position of terms are taken into account (unlike in BERT). With the transformer model, position and context were taken into account. However, the performance of BERT outshined the models due to the fact that it had been pre-trained on many instances of text data prior to being appended to the neural network. Despite the transformer model performing the worst out of the models we built, the accuracy is still slightly higher than the baseline of 0.57.

The results of the bag-of-words model can be summarized in the following table:

	Model 1: Unigram Text	Model 2: Bigrams Keyword + Text
Training Accuracy	0.96	0.95
Validation Accuracy	0.79	0.76

The chart below plots the training and validation accuracy across the number of epochs. This chart suggests that the bag-of-words model is overfitting, since there is a gap between the training and validation accuracy, where the validation accuracy starts to decrease as the training accuracy increases.



Model 1 refers to the bag-of-words model which utilizes unigrams and only the tweet text column (but not the keywords column). Thus, this model does not take into account the context of surrounding tokens nor the keywords related to the emergency. Model 2 employed bigrams instead of unigrams and also concatenated both the text and keywords columns to feed into the model. Thus, this model setup takes into account more context from the surrounding sentence (pairs of words instead of individual tokens) and also takes in more text overall (the emergency-related keywords). Both models were trained to produce the accuracy on the training set, and also run on the validation set to understand which combination of hyperparameters produced the highest accuracy.

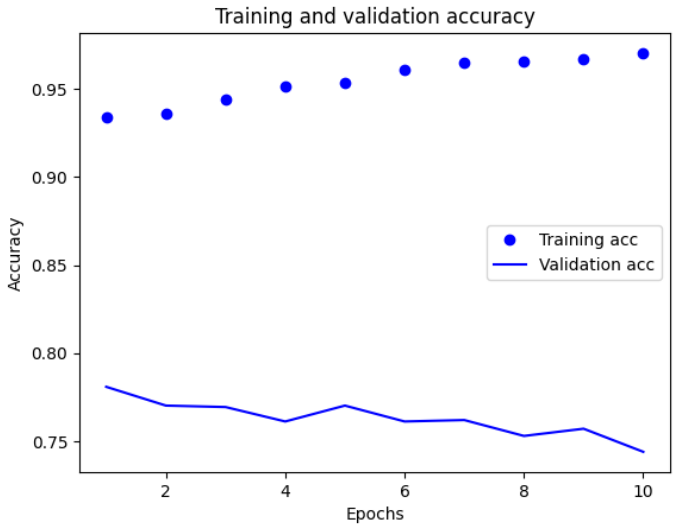
Other experimental alternatives were applied in order to see if they would add to the model's predictive power. For example, 16 hidden units were tried instead of 8, but this resulted in a validation accuracy of 0.74 (not as high as that of Model 1). This drop in validation accuracy is most likely due to overfitting which can result from adding more nodes per hidden layer. Another experimental tuning which we used was running the model for 20 epochs instead of just 10. This gave 0.77 accuracy on the validation set, which is also smaller than the 0.79 of Model 1. Thus, neither tuning technique was employed and instead the original parameters were picked when deciding which bag-of-words model was optimal.

GloVe Results

The GloVe embedding model is a type of unsupervised learning model that generates word embeddings by analyzing the co-occurrence statistics of words in a text corpus. In this case, it was trained to predict natural disaster tweet authenticity. With a training accuracy of 94% and a validation accuracy of 78%, this model shows potential for capturing the semantic meaning and context of words related to natural disasters. The high training accuracy suggests that the model has effectively learned the underlying patterns in the training data, while the decent validation accuracy suggests that it is able to generalize to new, unseen data.

However, it is worth noting that the validation accuracy is still not as high as desired, indicating that the model may not be capturing all of the relevant features needed to accurately classify the tweets. This could be due to a number of reasons, such as a lack of diverse and representative training data, the inability of the model to capture the nuances and complexities of natural disaster tweets, or suboptimal hyperparameters. Further fine-tuning and optimization of the model could potentially improve its performance and make it more effective for predicting the authenticity of natural disaster tweets.

	GloVe Model Performance
Training Accuracy	0.94
Validation Accuracy	0.78



HODL Transformer Results

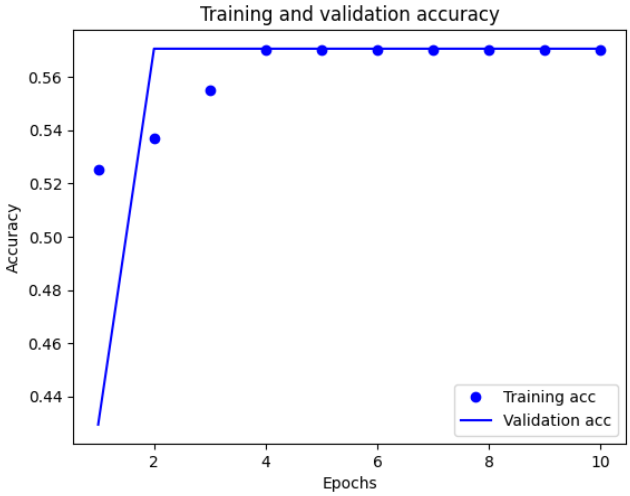
The transformer surprisingly gave the worst results, with a training and validation accuracy of 0.57, implying the model simply ranked all tweets as "not emergency" and applied the No Information naive model. This was unexpected, as transformers take into account both context and positional encoding. Perhaps this bad performance is due to the low sample size and no pre-training weights. In class, their power with respect to self-attention was discussed. Using self-attention heads, contextualized versions of the individual embeddings are produced and provide more information to classify tweets on (in the context of this project). Thus, it would be expected that the out-of-sample performance would at least be higher than that of the bag-of-words model (not context-dependent), but this was evidently not the case.

	Transformer Performance
Training Accuracy	0.57
Validation Accuracy	0.57

The chart below shows that the gap between the train and validation accuracies is small, which shows that the HODL Transformer model is not overfitting as much as the previous models

Future investigations into why this might have happened could involve experimenting with batch size and number of epochs for the transformer. Additionally, one could increase the number of self-attention heads employed, so as to learn even more patterns about the data. Each head picks up different information, and thus this might increase prediction power for the model. Finally, feeding in the

keyword along with the text instead of the text corpus alone might provide more information for the transformer to learn from.



BERT Results

Using transfer learning with BERT improved upon our bag-of-words models, as demonstrated in the table below:

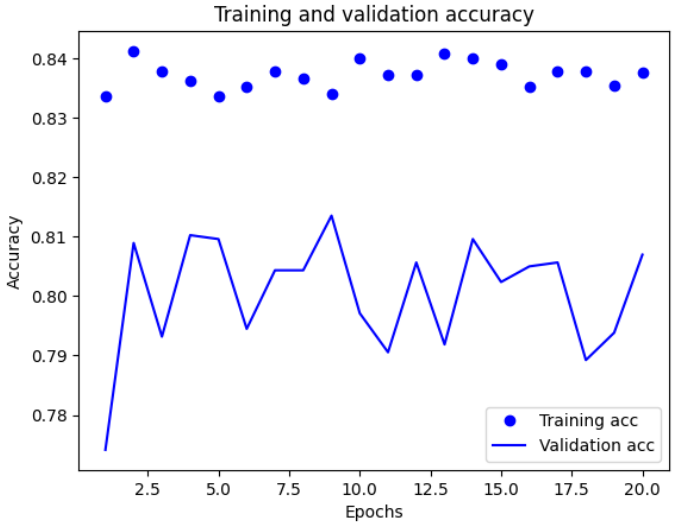
	Model 1: BERT Transfer Learning with 1 Dense Layer	Model 2: BERT Transfer Learning with 2 Dense Layers
Training Accuracy	0.83	0.84
Validation Accuracy	0.816	0.814

The chart below plots the training and validation accuracy as the number of epochs increases. This chart shows a relatively smaller gap between the train and validation accuracy, which suggests that the BERT model is overfitting less than the bag-of-words model.

BERT transfer learning was the best performing model in our case which makes intuitive sense for the following reasons:

- Contextual understanding: BERT is a contextual language model, which means that it is able to understand the meaning of a word or phrase based on its context within a sentence or document. In the case of tweets, which are typically short and contain limited context, BERT's ability to understand the context of each word can be especially useful in accurately classifying the tweet as emergency-related or not.
- Pre-training on large amounts of text: BERT has been pre-trained on a large corpus of text from the web, including social media text. As a result, BERT may be better equipped to handle the informal language, slang, and abbreviations that are often used in tweets. Transfer learning: BERT is designed to be fine-tuned on a wide range of downstream NLP tasks, including text classification.

As the BERT model was the best performing model when run on the validation set, it was also run on the test set to see how it would perform out of sample. The out-of-sample test set accuracy for the BERT model was 0.81.



Stacked Bag-of-Words and BERT

This model is a neural network ensemble model that takes two types of input (BOW and BERT) and predicts the authenticity of tweets regarding natural disasters. With an accuracy of 96% on the training set and 75% on the validation set, this model shows promising results in distinguishing between real and fake tweets. By leveraging the strengths of both the BOW and BERT models, this ensemble model is able to effectively capture the meaning and context of the tweets and make more accurate predictions.

However, it is worth noting that this model's performance was lower than that of the BERT model, which achieved an accuracy of 82%. This could be due to a number of reasons. For example, it is possible that the BERT model was able to better capture the nuances and complexities of natural disaster tweets due to its pre-trained language representations. Nonetheless, this ensemble model provides a valuable approach for incorporating multiple sources of information and could potentially be further improved with additional fine-tuning and optimization.

	Stacked Bag-of-Words and BERT Model Performance
Training Accuracy	0.96
Validation Accuracy	0.75

Lessons Learned

Our problem was to classify whether a tweet is a real emergency or not, and our best model was using transfer learning with BERT. This makes sense for a few reasons. First, the main problem of classifying whether a tweet is a real emergency is that tweets often contain words that have double meanings or slang. Especially on Twitter, people are likely to casually say words that would indicate an emergency, such as with the sentence "This movie was fire!" A human will interpret this as a normal way of saying the movie was good, but a machine learning model would not be able to know this without taking the other words in the sentence into context. Since BERT is a contextual language model, this helps address the problem of words having double meanings in tweets.

Second, the content of tweets could be anything. People use their personal twitter accounts to tweet about anything that is on their mind, ranging from topics such as movie reviews, politics, or memes. Twitter is also used, however, by organizations for official announcements, such as a disaster or emergency. Individuals may also tweet about an emergency in their area, which can trigger an organization to respond and send help. Since there is no limit to the domain of text that tweets can contain, BERT is helpful in capturing many examples of human language in various domains. It catches examples of words having multiple meanings, and even slang since BERT was also trained on social media text.

Third, tweets have a character limit and are often short chunks of text. A pre-trained model such as BERT is ideal for this problem, since our training dataset of tweets may not contain enough text to train the model on the various domains and meanings of the text within the tweets.

Finally, a lesson can be learned regarding the relationship between the number of tunable parameters in the model and the performance as observed through accuracy. The worst-performing model (HOLD Transformer) also had the most parameters (293,111,014). This model is very "complex", but it failed to even outperform Bag-of-Words. Often, this means overfitting could easily happen, and the model will have low out-of-sample performance, as is the case in this project. One future alteration would be to find the optimal balance for the transformer model to maximize accuracy while avoiding overfitting.

Future Directions

If given the opportunity to further this project in the future, we would love to compare the performance of the bag-of-words, BERT, and HODL transformer models to that of a Recurrent Neural Network (RNN) such as LSTM. While these kinds of models were the industry standard for a number of years, the emergence of transformers allowed for more predictive power. More specifically, they ensured that the output for a given input remained the same, took into account the context of the surrounding tokens, and incorporated the position of words within sentences. However, a comparison to the RNN

would be helpful in really understanding how much improvement we have gained from transformers as a whole.

Another area of future work could be predicting the type of emergency that triggered the positives as well. Our framework only predicts a binary for whether a tweet indicates a real emergency or not, but a valuable addition to our model could be identifying whether the tweet is about a fire, natural disaster, or mass shooting to list a few. Emergency responders could be the fire department, police, medical professionals, or others depending on the situation, so a categorical output could serve to notify which emergency responders should act as soon as possible. This would make our classification model more effective in putting emergency responses into action.

Colab Links

Bag of words and BERT:
https://colab.research.google.com/drive/1yMxnjOiy5ibXBfEB9fm8jljv6LAGe5gp?usp=sharing

GloVe:
https://colab.research.google.com/drive/10Wi5_kDRy4jrvKbmWBjucaE578VpFWvf?usp=sharing

Transformer:
https://colab.research.google.com/drive/1FpW2WnfqpKjqBdKLUt2hBLPK3KfxIcPu?usp=sharing

Stacked model link:
https://colab.research.google.com/drive/1yMxnjOiy5ibXBfEB9fm8jljv6LAGe5gp?usp=sharing

BERT transfer learning + demo link:
https://colab.research.google.com/drive/1h7htpIxkWgHO4i37nqqcAWk56iwE0foe?usp=share_link

References

Howard, A., devrishi, Culliton, P., Guo, Y. 2019. Natural Language Processing with Disaster Tweets. Kaggle. https://kaggle.com/competitions/nlp-getting-started.

RapidSOS. 2015. Outcomes Quantifying the Impact of Emergency Response Times. https://cdn2.hubspot.net/hubfs/549701/Documents/RapidSOS_Outcomes_White_Paper_-_2015_4.pdf

Krikorian, R. 2013. "New Tweets per second record, and how!" Twitter Official Blog.

Purvis. Current State of Turnout Times. https://www.purvis.com/current-state-of-turnout-times/.