CS 4650 "Real or Not? NLP with Disaster Tweets" Project

Meha Kumar
Georgia Tech
mkumar77@gatech.edu

Brooke Miller
Georgia Tech
bmiller@gatech.edu

Amanda Schmitt
Georgia Tech
aschmitt8@gatech.edu

1 Abstract

Our project aims to classify tweets provided by the Kaggle competition "Real or Not? NLP with Disaster Tweets" (kag, 2020) as either a real disaster or not a real disaster. We have implemented two baseline classifiers (Naive Bayes and Maximum Entropy) and three other classifiers (LSTM, an ensemble classifier, and BERT) to accomplish this task. The Naive Bayes model attained an F1 score of 0.78629, the Max Entropy model attained an F1 score of 0.78016, the LSTM model attained and F1 score of 0.79550, the BiLSTM model achieved an F1 score of 0.81186, the ensemble classifier attained an F1 score of 0.79856 and BERT achieved an F1 score of 0.79732. Overall, we met our goal achieving an F1 score of greater than 0.80, but there is still a lot of room for improvement in our implementation and tuning of these models. Our code for this project is hosted on GitHub at the following repository link: https://github.gatech.edu/bmiller80/NLP-Group22 The repository has a README for how to run the models contained on each branch.

2 Introduction and Related Work

In today's digitally enabled world, natural disasters and other emergencies can show up on social media before reaching authorities and aid teams. Monitoring tweets for disaster-related content can therefore be a useful tool for prompt emergency response. Text classification can be used as a simple tool for identifying whether a tweet is about a disaster. However, it can be unclear to an algorithm if a tweet is actually indicating a disaster, leading to incorrectly classified tweets. For example, the following tweet could be incorrectly classified as about a disaster, due to its use of the word 'ablaze' :

on plus side LOOK AT THE SKY LAST NIGHT IT WAS ABLAZE

(kag, 2020). This incorrect classification could lead to wasted resources as well as the spread of misinformation.

An example of previous work that has been done is the classification of situational awareness of tweets (Verma et al., 2011). "Situationally aware" tweets are those that are typically subjective, impersonal, and use formal register. This study created a classifier that incorporated these dimensions to determine whether tweets were situationally aware, with the motivation that these tweets are most important to both the public and authorities in terms of disseminating information. Situational awareness is closely related to disaster classification and so this work encourages further exploration into the domain of applying text classification techniques to tweets during mass emergency. Another work related to tweet classification in disaster scenarios implemented BERT to harness the power of user generated content (like tweets) to provide information about real time disasters (Ma).

Our main project goal is to create a submission for the Kaggle competition "Real or Not? NLP with Disaster Tweets". The aim of this competition is to analyze tweets from the given data set and to classify them as announcing a disaster or not. The competition is judged based on the *F1* score:

$$F1 = 2 * \frac{(precision * recall)}{(precision + recall)}$$

Our initial goal was to beat half of the existing teams by attaining an F1 score of 0.8 or higher.

Many options are available to develop a binary classifier. We will use Naive-Bayes and Maximum Entropy as our baseline models. Then, we will develop a basic LSTM and a bidirectional LSTM with max pooling, which has been proven to work better than other methods (Zhou et al., 2016). Based

on the work done by Ma and at the suggestion of our TAs, we will also implement BERT (Ma). We will also try an ensemble classifier composed of the Naive Bayes model, the Maximum Entropy model, and the bidirectional LSTM with max pooling model, and BERT. We want to try an ensemble classifier because (Kanakaraj and Guddeti, 2015) suggest that it may increase performance compared to individual classifiers. This type of classifier was also less popular among existing submissions, and we could not find one incorporating an LSTM.

This project has important real-world applications to better understanding actual tweets from times of disaster. The knowledge gained could be used by authorities during a disaster to determine which tweets are informative and save lives, without wasting a lot of time manually reading through tweets. It could also be used by social media applications like twitter to "verify" informative tweets, to both highlight relevant information as well as to prevent the spread of misinformation during emergencies.

3 Methods

3.1 Data

Our data has not changed since the midway report, but we have some additional figures. The data is provided in csv format by Kaggle for the competition. There is a training csv file and a testing csv file. Each row of data includes the ID number, a keyword, the location, and the text of the tweet. The keyword and location are not always provided. Below are some examples of tweets.

	id	keyword	location	text	target
15	23	NaN	NaN	What's up man?	0
16	24	NaN	NaN	I love fruits	0
17	25	NaN	NaN	Summer is lovely	0
18	26	NaN	NaN	My car is so fast	0
19	28	NaN	NaN	What a gooooooooaaaaaal!!!!!!	0

Figure 1: Non-disaster tweet examples.

Similar to the "ablaze" example in the introduction, the following non-disaster tweet is an example of how some tweets make classifying disaster tweets difficult, since the presence of keywords related to disasters does not necessarily mean that the tweet is about a disaster. In this case, one would think that the presence of the word "ambulance" may signal that the tweet is disaster related, but this may not always be true:

> People who try to j-walk while an ambulance is passing... I hate you.

On the other hand, here is an example of a disaster tweet that contains the same keyword:

> Twelve feared killed in Pakistani air ambulance helicopter crash

Simply identifying keywords is therefore not enough to build a classifier, which is why this problem is difficult and more NLP classifier options need to be explored.

	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1

Figure 2: Disaster tweet examples.

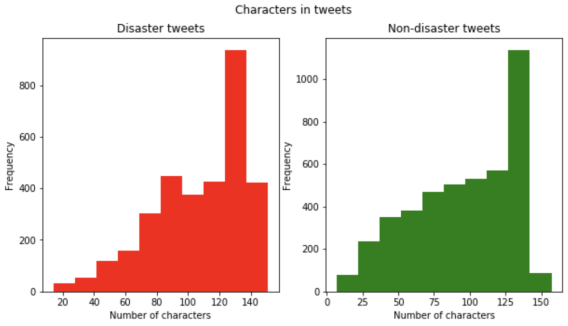

Figure 3: Plot showing number of characters in tweets before cleaning.

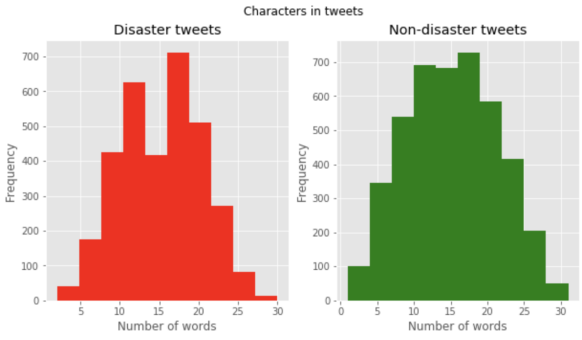

Figure 4: Plot showing number of words in tweets before cleaning.

Since the data come from tweets, there are many extraneous characters that do not contribute to the

meaning. The data was cleaned by removing URLs, extra white space, ellipses, and special characters. Additionally, contractions were cleaned, typos were fixed, and all text was lower-cased. The data cleaning process used was posted by other competition participants (Gunes, 2020; Nanto P, 2020). We performed a word count after cleaning. We also show the top ten words from each label after cleaning.

Plotting information about the data (which we learned is called Exploratory Data Analysis, or EDA) helps understand what type of cleaning or analysis will help the task. We learned through our plots and those of others that a lot of cleanup was needed to fix spelling, white-space, URLs, and special characters (Nanto P, 2020; Gunes, 2020). From printing out a few tweets, we thought the tweets were too messy because of these features. We also learned that hashtags did not seem very useful, since there was a lot of overlap between the positive and negative tweets' hashtags (Doomdiskday, 2020).

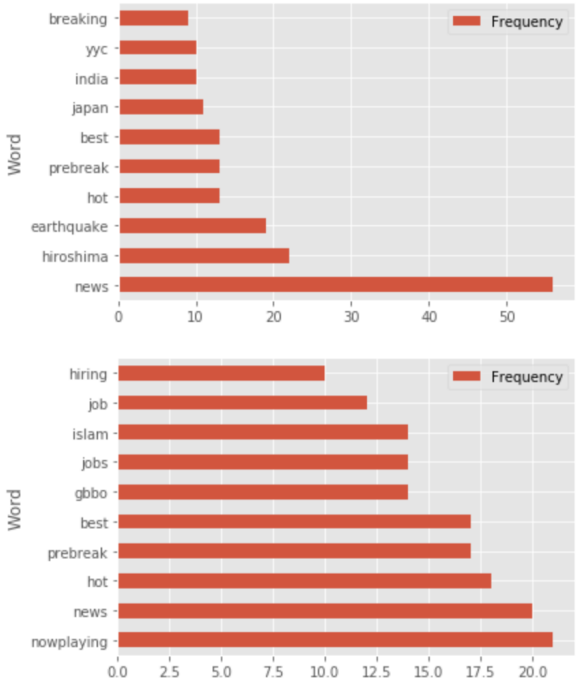

Figure 5: Most common hashtag words. Disaster tweets (top) and non-disaster tweets (bottom).

Data	# rows	# unique words
Train	7613	17092
Test	3263	10320

Throughout the course of the project, we have not changed the data used, but we did split it up into separate subsections, as will be explained in the models section.

3.2 Models and Analysis

The following sections describe each of the models implemented, including the baseline models that we compared them to.

3.2.1 Ensemble Classifier

The ensemble classifier model implemented is a combination of all of the other models. It uses two main features of an ensemble classifier: bagging and weighted polling. Bagging refers to the random selection of data sets (with replacement) to generate different models. To implement this, we tried multiple numbers and decided that three subsets, each half of the size of the full training data size will provide enough data for each individual model to attain a good accuracy, while avoiding over fitting from using too much of the data set. In development, we also took out 20% of the data to do an 80/20 train/test split. For the final submission, however, the data subsets were sampled from all of the training data (with no test subset removed). Weighted polling allows the predictions of each model to cast a vote towards the overall decision. After experimenting with different numbers and weights for each model, the following format was used. The models used for voting consisted of three Naive Bayes models each trained on a different subset of half the training data, three Maximum Entropy models each trained on a different subset of half the data, three LSTM models each trained on a different subset of half of the data, and 3 identical BERT models trained on all of the train data. This gives a total of 12 models polled. Additionally, based on the results that the LSTM model outperformed the others, the LSTM predictions were weighted twice as heavily, getting two votes each, leading to a total of 15 possible votes for each prediction. After testing different threshold values, the best performing was a majority rule, with an instance earning a true label if and only if it earned at least 8 of the 15 possible votes.

3.2.2 LSTM

Using the 200 dimension Twitter word embeddings from GloVe, we trained a preliminary LSTM with one hidden layer of 300 dimensions and a dropout rate of 0.5 for 50 epochs. The GloVe embeddings were chosen because they are a popular choice. A bidirectional LSTM with pooling

was run next, with improved results. A bidirectional LSTM seemed more appropriate because using knowledge from the beginning and end of a tweet may help the classifier decide. Max pooling helps because including only pooling along one dimension of a sample's matrix ignore the fact that the dimensions are not independent (Zhou et al., 2016). These modifications are known to increase performance of LSTMs. They are also known to learn faster than traditional LSTMs, and indeed we observed the bidirectional LSTM with pooling was able to exit without finishing all the epochs. The LSTM code used was from (Nanto P, 2020), with some modifications to the text cleaning.
Our LSTM model dealt with unknown words (as in, words that GloVe was not aware of) by adding an entry to our embeddings matrix for unknown words. This entry was filled with random numbers. Both LSTMs used an ADAM optimizer, which is designed for deep learning (Kingma and Ba, 2014).

3.2.3 BERT

BERT, Bidirectional Encoder Representations from Transformers, is a relatively recent tool developed for natural language processing (Devlin et al., 2018). BERT learns the relationships between words and how they tend to appear near each other. Unlike previous models, BERT looks at the context of any given word from both sides instead of just the words that appear before or just the words that appear after. BERT can be used for classification, next sentence prediction, question answering, named entity prediction, and more tasks, but we apply it here for binary classification. Our model was adapted from (Rajapakse, 2020) and fine-tuned a pretrained BERT model using the training set.

3.3 Baseline Models

3.3.1 Naive Bayes

Naive Bayes classifiers are a common choice for text classification because of their simplicity. The classifier utilizes Bayes rule to compute the probability of each class given the features present in the text and then selects the class with the highest probability. The probability of each class is computed using counts of the features' occurrences with texts of each class in the labeled training set.

Our Naive Bayes model was largely pulled from our homework assignment in the Georgia Tech Natural Language Processing class. The model uses a unigram feature extractor, that makes every word in the data set an element of the feature vector. We combine the cleaned tweet text and the optionally provided keyword into a string and then apply the unigram feature extractor.

3.3.2 Maximum Entropy

Maximum Entropy is another common probabilistic model that uses linear regression to identify the most likely label for a class. Just as in our Naive Bayes model, we use the same unigram feature extractor applied to a string containing the provided key word and tweet text.

4 Results

4.1 Experimental Setup

The following sections outline the details of hyperparameters used as well as train/test split and similar protocols for each of the models used. After training and tuning various parameters to achieve the highest possible accuracy, we formatted the predictions of the models on the actual test set and submitted them to the Kaggle competition. This is where the F1 scores reported come from, as we did not have access to the actual labels of the test set.

4.1.1 Naive Bayes

The model is trained on the full train data set and then applied to the test set.

4.1.2 Maximum Entropy

The model is trained on the full train data set and then applied to the test set, using the hyper parameters below.

Hyperparameter	Value
Learning Rate	0.2
Iteration	5000

4.1.3 LSTM

The LSTM models were both trained by splitting the existing training data into an 80/20 split for training and development data, respectively. Below is a table of the hyperparameters used.

Hyperparameter	Value
Learning Rate	0.001
Maximum Epochs	50

4.1.4 Ensemble Classifier

The ensemble classifier used the relevant hyperparameters and train/test split as the individual models that comprised it, with the exception of Naive Bayes and Maximum entropy. Since these had the two lowest scores, they were modified to not

include intermediate testing sets (and instead sampled the whole train data set). As expected, the ensemble classifier outperformed all four of its constituents, even if only by a small amount. This is the goal of the ensemble classifier: in searching for a majority decision on the classification, the combination of all of the models can make up for the places where an individual model may be lacking.

4.1.5 BERT

The BERT was trained using an 80/20 split of the training data to create a train and a development set. It used the following hyperparameters

Hyperparameter	Value
Training Batch Size	24
Learning Rate	0.00002
Training Epochs	1

4.2 Result Comparison

We evaluated all our models using the same test data set provided by Kaggle. We used the F1 score described earlier to measure them against each other. Below is a table of the F1 scores accomplished.

Model	Test F1 Score
Maximum Entropy	0.78016
Naive Bayes	0.78629
BERT	0.78732
LSTM	0.79550
Ensemble Classifier	0.79856
Bidirectional LSTM with pooling	0.81186

In the Kaggle competition, we found that out of the 2632 teams with submissions, only ten of them had scores above 0.90 but less than 1. A small note on comparing our submissions to the posted submissions is that there were many submissions that received 100% due to the correct labels for the test data set being leaked, which is why we are ignoring the models with 100% accuracy in our comparison, since it is very likely that these people cheated by using the actual test labels. When researching about other approaches to this problem, we found the information about the test labels being leaked, and of course did not use this illegal data, which is why we were ranked lower compared to the many "perfect" submissions. The fact that there are a few hundred perfect submissions and only 10 in the 90 range also demonstrates that these are not reliable. Therefore we feel that, as beginners, we performed fairly well. We basically met our goal to reach an F1 score of 0.80 on most models, and exceeded our goal with the BiLSTM with pooling.

Our baseline models were Naive Bayes and Maximum Entropy, and our LSTM and BERT models outperformed them. The other BERT notebooks available appear to all reach about a 0.83 F1 score on Kaggle, so we are confused as to why our model performed lower when it ought to have been the best model. Our bidirectional LSTM with pooling performed about as well as other LSTM notebooks on Kaggle, which hovered around 0.80. We would have expected them to perform even higher than the baseline models, but it seems that for this task, without more specific modifications, this is the limit reached.

In terms of ablation, we did compare our LSTM model to a bidirectional LSTM with pooling. We only changed the model architecture, not the hyperparameters. The bidirectional LSTM ended up being slightly better.

Our BERT model did not perform as well as we thought it should. This could be due to the fact that we were only able to train our BERT model for one epoch, since we do not have GPUs. If we had been able to run BERT freely, we could have optimized hyperparameters including epoch number.

The following are confusion matrices for the various models. Though we did not use the leaked test labels at all during development, we did use them to generate these confusion matrices to compare the ratios of false positives, false negatives, true positives, and true negatives across our models.

Confusion matrix for Naive Bayes

	Predicted No	Predicted Yes
No	1539	322
Yes	380	1022

Confusion matrix for Maximum Entropy

	Predicted No	Predicted Yes
No	1525	336
Yes	360	1042

Confusion matrix for LSTM

	Predicted No	Predicted Yes
No	1638	223
Yes	424	978

Confusion matrix for ensemble

	Predicted No	Predicted Yes
No	1671	190
Yes	424	978

Confusion matrix for BiLSTM

	Predicted No	Predicted Yes
No	1638	223
Yes	392	1010

Confusion matrix on BERT

	Predicted No	Predicted Yes
No	1549	312
Yes	338	1046

As far as analyzing *why* these were misclassified, previous research found that tweets for disasters are often misclassified because they are too short, are sarcastic, are ambiguous, or talk about potential disasters rather than actual ones (Ma). Based on this previous research as well as the other scores submitted on the Kaggle leader board, it is safe to say that the disaster classification of tweets using NLP is a difficult task. Below we highlight a few misclassified tweets to demonstrate the types of tweets we misclassified with the best performing model, the BiLSTM model (many of these tweets were also likely misclassified by the other models as well).

The following tweets are false positives, meaning our model thought they were genuine disasters but they are not.

Some tweets seem to be about genuine medical advice, but mention possibly confusing words like ”death”, ”destroy”, and ”blood”.

> HEALTH FACT: Women account for 39% of smoking deaths. http://t.co/fSx2H9XAkI

Some tweets discuss hypothetical or fictional disasters.

> @MalikChaimaa I hope Zayn gets blown up in a drone attack whilst visiting family in Pakistan ??

> This week on dream from last night: my mom drove my car in a snowstorm. She hit another car while parallel parking. Saw my first ex.....

> She rose to her lofty position after being transported by accident to Oz in a hot air balloon during a snowstorm.

Even when the tweet is clearly discussing a disaster, it can be difficult for the classifier to distinguish between personal disasters and community disasters.

> I deserve a goddamn medal for dealing with this basement flood without panicking

Some tweets use flowery language in metaphor; we found a few poems and song lyrics. Such tweets might confuse a classifier because they genuinely sound distraught, but a human reader would recognize it’s poetry.

> deluged dismay so
> soon surrendered summer drought
> descending disturbed
>
> #sunrise #haiku #poetry #wimberley #atx#smtx http://t.co/z9FDebg5Fm

These misclassified tweets above are all examples of false positives, where our classifier thought that a non-disaster tweet was a disaster tweet. There were also many instances where the classifier thought a disaster tweet was a disaster tweet, the false negatives.

However, when looking through false negatives, we also saw many tweets that were mislabelled in the first place. For example, this tweet is clearly not about an actual disaster:

> Burning bridges is my forte!

And neither is this one

> I suffered a catastrophic pants failure at work today. I resorted to tying my jacket around my waist til I got to a thrift store. trustory

Our text cleaning could have been a reason for this. The tweets shown above are the original tweets, but our model judged tweets after cleaning. Despite this possibility, there is a real concern that some of these ground truth labels may not be accurate.

> Would you blow me kisses If I kept my distance? Would you send a hurricane As proof of your existence?

On the false negative side, we have another example of song lyrics. This is labeled as a disaster but our model does not recognize it as being about a real hurricane. We struggle to recognize and correctly label humor when it about an actual disaster,

even though making humorous comments about serious disasters is something that frequently happens on twitter.

4.3 Work Division

Data cleaning was a collaborative effort as all of us found useful cleaning methods in other notebooks. Amanda was responsible for the Ensemble Classifier. Brooke was responsible for the Naive Bayes Classifier, Maximum Entropy Classifier, and BERT Classifier. Meha was responsible for the LSTM models. We all worked together to write and edit the proposal, midterm report, and final report. We also frequently discussed results and helped each other troubleshoot various issues throughout the process of development and analysis.

5 Conclusion

We found that it is difficult to improve the task of classifying possible disaster tweets. Our bidirectional LSTM model worked the best, but further improvements are needed. We also confirmed that Ensemble classifiers can give results better than each of their individual models.

Compared to other Kaggle submissions on the same task, we feel we matched many of the teams which used the same models. Our Ensemble classifier did not end up being the star of the show, as we had hoped, but we still achieved decent F1 scores. Our Ensemble classifier was the main new contribution we had to this task.

To improve upon this work, we could relabel the data set because it became clear that there are many mislabelled examples. We also found several duplicates with two different labels, so this issue needs more attention as well. These double-labelled examples would need to be resolved by hand, which may prove difficult since some tweets may be ambiguous even to humans.

After this, we could perform more EDA to see if there is any further (or less) cleaning we could do to improve performance. Tweets are an especially tough text to handle because there are lots of misspellings and abbreviations that are difficult to identify and fix. We cleaned quite a bit, but perhaps misspelled words, contractions, and the presence of URLs could actually help the model determine whether there is a disaster. We could add meta-features specific to this task to enrich the data. Possibly informative features could be the length of a tweet in characters or words, the number of URLs, the number of hashtags, the number of emojis, the number of out-of-vocabulary words, and the presence of an image. All of these features could be investigated through EDA to observe visually their significance before employing them in the model.

Another possible improvement would be to incorporate computer vision to identify and label images included in the tweet. Our first example,

> on plus side LOOK AT THE SKY LAST NIGHT IT WAS ABLAZE

included a picture of a sunset in the original tweet. This may greatly help classify tweets with images in their URLs. Some computer vision software can now label what is in an image fairly comprehensibly, and including those words as an additional feature could help.

For our models, we could train our models more to improve hyper-parameters. As it stands, none of us own GPUs, so our ability to run LSTM and BERT were especially limited. When training BERT, we only ran one epoch, so training longer would certainly improve it's performance. The Ensemble classifier could poll more instances of the models as well to see if that helps, and weight them all according to their respective accuracy, rather than just weighting the best one with twice as many votes.

Overall, we learned a lot from implementing and comparing these models on this difficult task of classification of disaster tweets, but there are still many improvements that could be made on these strategies.

References

2020. Real or not? nlp with disaster tweets.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Doomdiskday. 2020. [full tutoria] eda to ensembles. embeddings zoo!

Evitan Gunes. 2020. Nlp with disaster tweets - eda, cleaning and bert.

M. Kanakaraj and R. M. R. Guddeti. 2015. Nlp based sentiment analysis on twitter data using ensemble classifiers. In *2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, pages 1–5.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Guoqin Ma. Tweets classification with bert in the field of disaster management.

Leksono Nanto P. 2020. Eda ml deep learning with sklearn and pytorch.

Thilina Rajapakse. 2020. A simple guide on using bert for text classification.

Sudha Verma, Sarah Vieweg, William Corvey, Leysia Palen, James Martin, Martha Palmer, Aaron Schram, and Kenneth Anderson. 2011. Natural language processing to the rescue? extracting "situational awareness" tweets during mass emergency.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*.