Predicting the success of global terrorist activities

Trisha Jani (trishaj@stanford.edu)

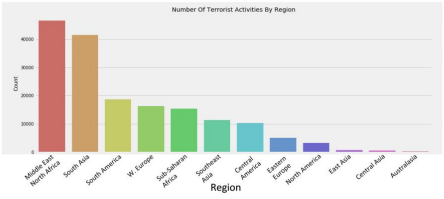
Background

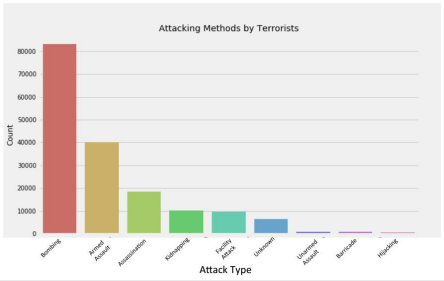
Since the turn of the century, we have seen an uptick in the number of terrorist activities on a global scale. We aim to better understand terrorist activities by using deep learning to predict the success of attacks. Such analysis serves several purposes:

- Better understand terrorist preferences/patterns
- Prepare/prevent potential attacks, saving lives
- Save resources if attack's success seems unlikely

Dataset & Feature visualization

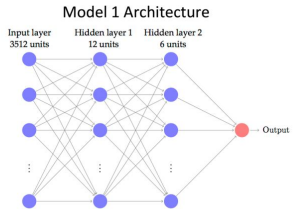
The initial dataset consisted of 170,350 terrorist incidents described by a list of 135 features Each incident is labeled as successful or unsuccessful. There is a data imbalance: 90% of incidents are marked successful. We performed basic exploratory data analysis to to gain a high level understanding of the our features. Our final models make use of 9 features: `region, country, attack type, target type, weapon type, group, suicide, multiple, hostage.`

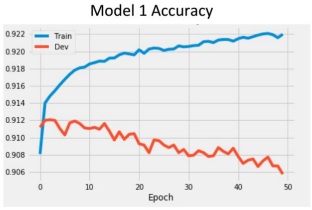




Neural Network Architectures

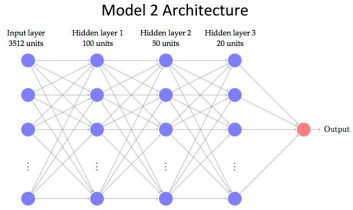
Model 1: Fully Connected NN with two hidden layers

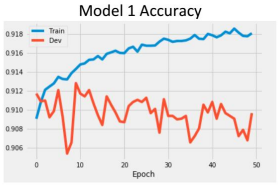




After tuning, we found the following parameter values led to the highest accuracy:
Adam optimizer, α=0.001, ReLU for hidden layers, sigmoid for output layer
We trained our model for 50 epochs. Our dev set accuracy seems to decrease.

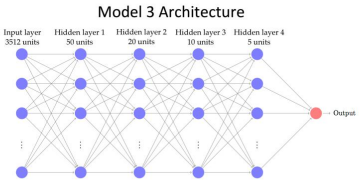
Model 2: Fully Connected NN with three hidden layers

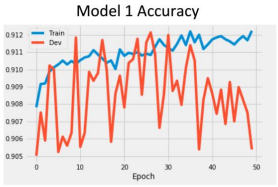




After tuning, we found the following parameter values led to the highest accuracy:
Adam optimizer, α=0.01, ReLU for hidden layers, sigmoid for output layer, balanced class weights, dropout for hidden layers (0.5, 0.2, 0.2)
We trained our model for 50 epochs. Our dev set accuracy follows a downwards trend.

Model 3: Fully Connected NN with four hidden layers





After tuning, we found the following parameter values led to the highest accuracy:
Adam optimizer, α=0.01, ReLU for hidden layers, sigmoid for output layer, balanced class weights, dropout for hidden layers (0.5, 0.2, 0.2, 0.2)
We trained our model for 50 epochs. Our dev set accuracy does not drastically increase.

Results

Table 1. Model Accuracy Comparison

	Train	Dev	Test
Model 1	92.34%	90.58%	90.85%
Model 2	91.98%	91.18%	91.17%
Model 3	90.83%	90.54%	90.69%

After cleaning our dataset, we had a a total of 151,254 points. Our training set had 122,515 points, our dev set had 13,613 points, and our test set had 15,126 points.

Table 2. Statistical Performance on test set

	Precision	Recall	F1 Score
Model 1	0.921	0.980	0.950
Model 2	0.916	0.994	0.953
Model 3	0.905	0.999	0.950

Conclusions

- Model 1, the simplest NN, has the lowest training error, but does not generalize as well
- Model 2, the fully connected NN with three hidden layers has highest performance on test set, does not overfit too much
- Model 3, the most complex model, has lowest accuracy, but does not overfit
- Data imbalance makes this problem hard, and it is helpful to use weighted cross entropy loss
- Dropout helps with regularization

Future work

Given more time and compute power, we would:

- Explore with deeper architectures
- Try more regularization techniques, tune dropout parameters further
- Incorporate more features from original dataset
- Further tune the weighting parameter between the two classes
- Try a more real-world metric (need new data)