Identifying Toxic Comments through Deep Learning

David Cheng and Ahmad Khan

Objective: Can we automatically flag inappropriate comments on public forums using Deep Learning?

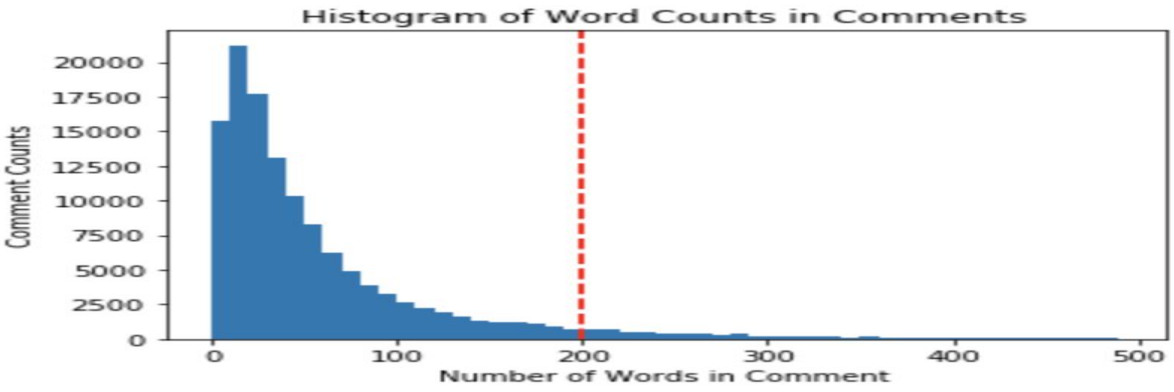
- Data pulled from Kaggle's Toxic Comment Classification Challenge
- 150K actual Wikipedia comments which were labeled for inappropriateness by human labelers
- 6 Classes of Comments (Toxic, Severe Toxic, Obscene, Threat, Insult, Identity Hate)
- Classes are not mutually exclusive (A comment can be flagged as multiple classes or even none of the 6)

Objective: Can we identify different types of toxic comments using Deep Learning?

Category (6 Classes)	Example Wikipedia Comment
toxic	“Ask Sityush to clean up his behavior than issue me nonsensical warnings..."
severe toxic	f**k you petty ugly ass desperate no life.. no status in society a*** retentive wiki admins (the site is great.. but the lower level admins have no life and can't handle the little authority they have.. it tells you how small and pathetic their lives are) if i get blocked ill be on in 20 seconds with a new ip or the pizza is free. cheers
obscene	Definitely have better things to do than read all that rules crap.... If I can't write it I can't write it no big whoop, it's not that important talk #c
threat	Regarding your passing Because you willfully violate Wikipedia's copyright and because you intentionally publish libel, I will arrange to have your life terminated.
insult	Because it is an R&B; album, and Wikipedia's choice in reliable sources is retarded.
identity hate	Why is your ******* so small? It is because you are ****?

Data Preprocessing

1) Removing Punctuations and converting all words to lower-case
2) Mapping each word in comment to a unique Integer (Keeping only top 250K words in Vocab) and passing integer values into the neural net (for embedding layer)
3) Limiting Comments to 200 words only based on comment length distribution shown below
4) Padding comments less than 200 words with 0's to the left of the first word



Baseline Model - Predicting (Majority Class) All Zeros

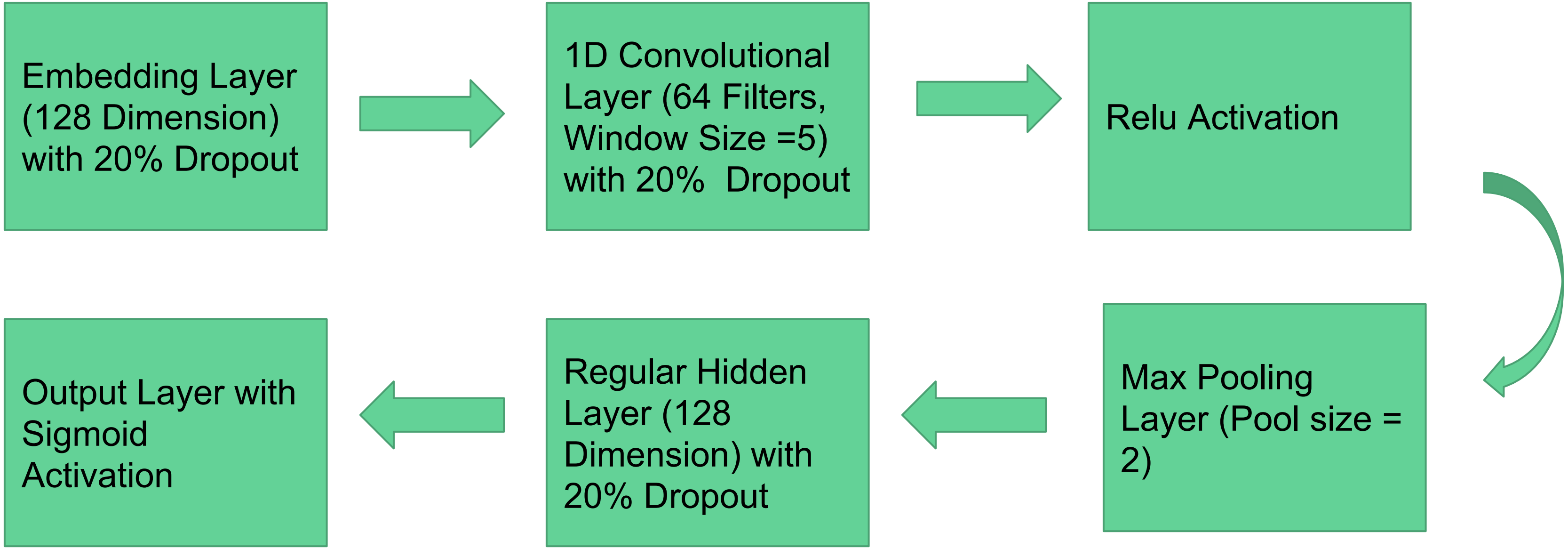
Comment Distribution and Baseline Accuracy for Each Class (Total 150K Comments in Dataset)

	Toxic	**Severe Toxic**	**Obscene**	**Threat**	**Insult**	**Identity Hate**
% of Comments	9.54	0.98	5.26	0.31	4.87	0.88
Class Accuracy %	90.46	99.02	94.74	99.69	95.13	99.12

Extremely Imbalanced Classes means that we can get a very high accuracy (96.3%) by simply predicting nothing as inappropriate despite having no precision or recall.

Need to train a model that not only improves accuracy but does so by achieving high precision and recall

Model 1: CNN Architecture

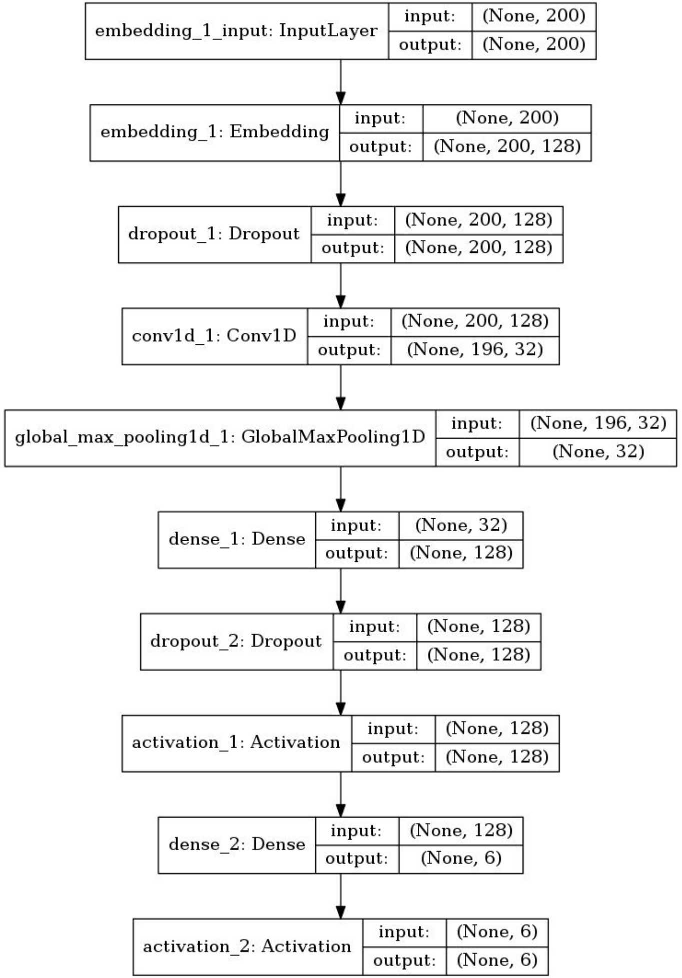


CNN Model Alt Visualization

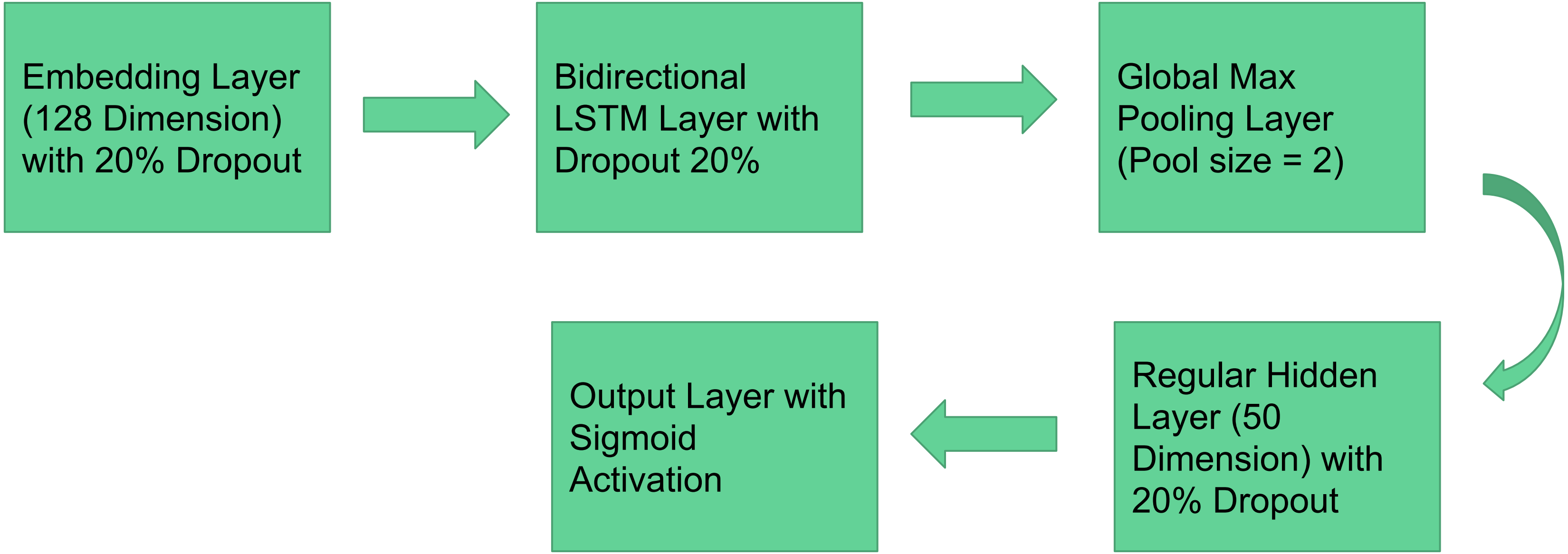
Generated by keras.utils.plot_model

Parameters of this particular model

Max_features =	200
Filters =	32
Stride =	1
Kernel_size =	5
Embedding-dim=	128
Batch Size =	64
Hidden Dim =	128
Convolved Layers =	1
Activation Function =	Relu
Loss Function =	Binary Crossentropy
Number of Epochs =	1



Model 2: LSTM Architecture



Model Evaluation

Model Performances on Validation Set

Model	Loss	Accuracy	Precision	Recall	Subset Accuracy
LSTM	0.048	0.9831	0.823	0.671	0.921
Final CNN	0.053	0.9821	0.750	0.731	0.912
Baseline	N/A	0.963	0.000	0.000	0.897

Precision and Recall using CNN for each class

	Toxic	Severe Toxic	Obscene	Threat	Insult	Identity Hate
Precision	0.740	0.529	0.809	0.00	0.736	0.808
Recall	0.813	0.424	0.822	0.00	0.702	0.131

Both LSTM and CNN models perform better than baseline across all metrics with the LSTM performing slightly better than the CNN

However, performance varies for each class...with some classes like Threat not being picked on at all

Architecture Tuning

Number of Convolutional and Max Pooling Layers	Train Accuracy	Validation Accuracy
1	0.9900	0.9814
2	0.9896	0.9782
3	0.9868	0.9769

Activation Function	Validation Loss	Validation Accuracy
Sigmoid	0.05680	0.9800
Relu	0.05536	0.9817
Tanh	0.05576	0.9816
Elu	0.0567	0.9816

Hyper Parameter Tuning

Model Rank	Filters	Kernel Size	Embedding Dimension	Hidden Dimension	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
1	64	5	128	128	0.0292	0.9888	0.0520	0.9821
2	128	2	128	128	0.0313	0.9876	0.0499	0.9820
3	64	4	128	128	0.0297	0.9883	0.0523	0.9820

(Above) Top 3 Model Parameter Choices

(Below) Bottom 3 Model Parameter Choices

Model Rank	Filters	Kernel Size	Embedding Dimension	Hidden Dimension	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
22	64	2	32	128	0.0395	0.9845	0.0517	0.9805
23	64	3	128	128	0.0311	0.9879	0.0524	0.9805
24	64	3	32	128	0.0359	0.9861	0.0537	0.9800

Note: the best model is only slightly better than the worst model hyperparameters in this set

Parameter tuning had limited gain for this dataset

Model Evaluation on Kaggle Test Set

On Kaggle test data both models performed similarly. However, by introducing blending/ensembling and taking the average of both the predictions superior performance was achieved

Model	**Test Score (AUC)**
Baseline	0.5000
CNN Model	0.9638
LSTM Model	0.9630
Blended Ensemble (50% CNN + 50% LSTM)	**0.9705**

Areas for Improvement

1) Using pre-trained word embedding models like word2vec and glove for getting better word embeddings
2) Using punctuation and upper case as unique words instead of removing them
3) Converting misspelled words into the most similar word in the corpus (using some sort of similarity metric like levinshtein distance)
4) Using ensembles of many more different models to blend in predictions from many uncorrelated models
5) Use AUC for training as well since the classes are so imbalanced making accuracy a less useful metric for model evaluation