Proportion Community Com	State		<pre>import pandas as pd import os for dirname, _, filenames in os.walk('/kaggle/input'): for filename in filenames: print(os.path.join(dirname, filename))</pre>
Comment First	Comment Comm		<pre>import pandas as pd from sklearn.model_selection import train_test_split from keras.preprocessing import text, sequence from keras.utils import pad_sequences from keras.models import Sequential from keras.layers import Embedding, SimpleRNN, LSTM, Spatial Dropout1D, GRU, Bidirectional, Input, Dense, Act;</pre>
The content of the	The content of the content and content of the con		from tqdm import tqdm
			<pre>try: tpu = tf.distribute.cluster_resolver.TPUClusterResolver() print('Running on TPU', tpu.master()) except ValueError:</pre>
Part	Testing		<pre>if tpu: tf.config.experimental_connect_to_cluster(tpu) tf.tpu.experimental.initialize_tpu_system(tpu) strategy = tf.distribute.experimental.TPUStrategy(tpu) else: #default distribution strategy in tensorflow, Works on CPU and single GPU</pre>
	Training		<pre>train = pd.read_csv('C:/Users/Richa/OneDrive/Desktop/TARP PROJECT_TOXIC COMMENT/dataset/jigsaw-toxic- validation = pd.read_csv('C:/Users/Richa/OneDrive/Desktop/TARP PROJECT_TOXIC COMMENT/dataset/validatase</pre>
B. COMPORTED	1		
Comparison Com	Control Description Desc		train.head()
train deep (Paeves, confet, "decement, "the color, "characte," "december, "de	traininteger (Personal and C. Manager, "Alternative Classics," Alternative Color patterns and Color of Colors and Color of Colors and Color patterns and Color patter	1	D 00000997932d777bf Explanation\nWhy the edits made under my usern 0 0 0 0 0 0 0 I 000103f0d9cfb60f D'aww! He matches this background colour I'm s 0 0 0 0 0 0
textion deeper("coverage_leasts", "between", "between", "security", "descritery paths"); and s = 1, implace = Tower of Administration of Coverage 1, implace	train drops ("covers_count", "absorbs", "absorbs", "assorbs", "assorbs", about any abo		
Decrease and the of common percent about a new chairs for received and have a stable for common percent about a common percent about a stable following the common percent about a stable following th	Process are the of commons been column to set that for contains a details processing and him is shadon's amount and 1 peoply (shadon's value of (a.e.gi 112)) associty process. Processing the shadon's amount and 1 peoply (shadon's value of (a.e.gi 112)) associty and the shadon's amount and 1 peoply (shadon's value of (a.e.gi 112)) associty and the shadon's amount and 1 peoply (shadon's value of (a.e.gi 112)) associty and the shadon's amount and the shadon'sh		
tentation, which, yearin, yearin = train_cost_stall(train_commont_cost_wales, train_cost_wales, strain_cost_wales, strain_cost_	### STATES Version Ver		<pre>#check max len of comment_text column to use this for padding in future pad_len = train['comment_text'].apply(lambda x:len(str(x).split())).max() print('max len of comment_text column',pad_len)</pre>
Identification Jen (1970-1975)	Identification Jen (avoid)		
cokenisation and Padding with max len of words in curpus Section Decoration	cokenisation and Padding with max len of words in curpus Section Debut We add with player 12 doctor date.		
id content lang Doco Who odit vid badger 12, occor our. I from expression, es note a sery edologramo. 2 Content sur de la ser de la quella ser de la content de la content la	id content lang DocorWho adh wie badgen 12. docordam.		
	Description		okenisation and Padding with max len of words in curpus
Doctor Who add will beginns 12 doctor dar tr 1 Storie sources, rea most a weary econograms. 2 2 Qualitation as a most a weary econograms. 3 3 Manual percelessified and shyle birsy tr 4 Seministrateopular percental style birsy tr 5 3 Manual percelessified and shyle birsy tr 5 4 Seministrateopular percental style birsy tr 5 5 Manual percelessified and shyle birsy tr 5 6 Manual percelessified and shyle birsy tr 5 7 Manual percelessified and shyle birsy tr 5 8 Manual percelessified and shyle birsy tr 5 9 Manual percelessified and shyle	Doctor Who add with bagingma 12 dector start. # # 1 Borne exposure, rep in our are mery reconstance. # # 2 Condition as to an equal content of the start of the		
<pre>xtent = best.content.values token.fit.on_text01listinrednn + list(xvalid) + list(xtest) xtent_seq = token.coxts_to_sequences(xtent) x_tent_peq = token.coxts_to_sequences(xtent) x_tent_peq = boken.coxts_to_sequences(xtent) x_tent_peq = boken.coxts_to_sequences(xtent) x_tent_ped = ped_sequences(x_tent_req,maxlen = max_len) x_tent_ped_sequences(x_tent_req,maxlen = max_len) x_tent_ped_sequences</pre>	<pre>xtent = best.content.values token.fitco_fitexts(listottes(n) + list(xvalis) + list(xtest)) x train_seq = coken.texts_to_sequences(xtrain) x valid_seq = coken.texts_to_sequences(xtrain) x valid_seq = coken.texts_to_sequences(xtrain) x_test_pag = token.texts_to_sequences(xtrain) x_test_pag = token.texts_to_sequences(xtrain) x_test_pag = pag_sequences(x_valid_seq.maxlen = max_len) x_train_pad = pag_sequences(x_valid_seq.maxlen = max_len) x_test_pad = pag_sequences(x_test_meq.maxlen = max_len) x_test_pad = pag_sequences(x_test_meq.test_med.test_meq.test_meq.test_meq.test_meq.test_meq.test_meq.test_med.test_meq.test_meq.test_med.test_med.test_med.test_med.test_med.test</pre>	4	#using keras tokenizer token = text.Tokenizer(num_words = None)
<pre>x_train_pad = pad_sequences(x_train_seq.maxlen = max_len) x_test_pad = pad_sequences(x_lest_seq.maxlen = max_len) x_test_pad = pad_sequences(x_lest_seq.maxlen) x_test_pad = x_test_pad_seq.maxlen = x_te</pre>	<pre>x_train_pad = pad_sequences(x_train_seq.maxien = max_len) x_teat_pad = pad_sequences(x_lens_seq.maxien = max_len) x_teat_pad_seq.maxien = part of cur datasets x_teat_pad_seq.maxien = pad_seq.maxien = max_len) x_teat_pad_seq.maxien = max_len) x_teat_pad_seq.maxien = pad_seq.maxien = pad_se</pre>		<pre>xtest = test.content.values token.fit_on_texts(list(xtrain) + list(xvalid) + list(xtest)) x_train_seq = token.texts_to_sequences(xtrain) x_valid_seq = token.texts_to_sequences(xvalid) x_test_seq = token.texts_to_sequences(xtest)</pre>
Classification based on GRU(Gated Recurrent Unit) Foreste an embedding metrics for the words which are part of our datasets embedding metrics = mineroes((len(word index) + 1,300)) For word, in tegmiword index Ltems(i); For word, in tegminal probability in the words of the model in the words of the model in the words of	Classification based on GRU(Gated Recurrent Unit) **Create an embedding metrics for the words which are part of our datasets embedding metrics = mp.tercs((ten(word index) + 1,300)) **For word, is in tenjetword index.ltmm(t); **For word, is not Mone: **The method of the method of the words which are part of our datasets embedding vector = embedding index.ltmm(t); **For word, is not word in the words in the wo		<pre>x_train_pad = pad_sequences(x_train_seq, maxlen = max_len) x_valid_pad = pad_sequences(x_valid_seq, maxlen = max_len)</pre>
Accesse on embedding metrics for the words which are pert of our detasets embedding metrics = np.veros((len(word index) + 1,300)) for word i is tapdaword index.timen()) for word i is tapdaword index.timen()) if embedding vector = macedding index.get(word) if embedding vector = macedding index.get(word) if embedding vector is not None: embedding metrics(i) = embedding_vector model = Sequential() model = Seq	foreste an embedding metrics for the words which are part of our datasets embedding metrics = mp.teros([den/word_index] + 1,300]) for word, is tagethered index items(); for word, is tagethered index items(); for words, is tagethered in our None; embedding vector = embedding_vector 0001		
<pre>embedding metrics = mp.zeros((len(word_index) + 1,300)) for word,i in eqdm(word_index).tess(): embedding vector = embedding_index.get(word) if embedding_vector is not None: erbedding_metrics(i) = embedding_vector 1001. </pre>	<pre>sebedding metrics = op.zeros([len(xord_index) + 1,300)) for word_i in tegdm(xord_index).tems(): embedding_vector = embedding_index.get(xord) if embedding_vector = embedding_index.get(xord) if embedding_vector is not None: embedding_setrice(i) = embedding_vector 1005.</pre>		
<pre>%time with strategy.scope(): model = Sequential() model.add(Embedding(len(word_index) + i,</pre>	<pre>%time with strategy.scope(): model = Sequential() model.add(Embedding(len(word_index) + i,</pre>		<pre>embedding_metrics = np.zeros((len(word_index) + 1,300)) for word,i in tqdm(word_index.items()): embedding_vector = embeddings_index.get(word) if embedding_vector is not None:</pre>
<pre>with strategy.scope(): model = Sequential() model.add(Embedding(len(xord_index) + 1,</pre>	<pre>with strategy.scope(): model = Sequential() model = Sequential() model = Add(Embedding(len(word_index) + 1,</pre>		
<pre>model.add(Embedding(len(word_index) + 1,</pre>	model.add(Embedding (len(word_index) + 1, 300, weights = [embedding_metrics], input_length = max_len, trainable = False)) model.add(SpatialDropoutD(0.3)) model.add(CRU(300)) model.add(CRU(300)) model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy']) model.summary() Model: "sequential_3" Layer (type)		
<pre>weights = [embedding_metrics], input_length = max len, trainable = False) model.add(SparialDropoutID(0.3)) model.add(Dense(1, activation = 'sigmoid')) model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy']) model.summary() Model: "sequential_3" Layer (type)</pre>	<pre>weights = [embedding_metrics], input_length = max len, trainable = False)) model.add(GRU(300)) model.add(Dense(1, activation = 'sigmoid')) model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy']) model.summary() Model: "sequential_3" Layer (type)</pre>		<pre>model.add(Embedding(len(word_index) + 1,</pre>
model.add(Dense(1, activation = 'sigmoid')) model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy']) model.summary() Model: "sequential_3" Layer (type)	model.add(Bans(1, activation = 'sigmoid')) model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy']) model.summary() Model: "sequential_3" Layer (type)		<pre>weights = [embedding_metrics], input_length = max_len, trainable = False))</pre>
<pre>model.summary() Model: "sequential_3" Layer (type)</pre>	<pre>Model: "sequential_3" Layer (type)</pre>		<pre>model.add(SpatialDropout1D(0.3)) model.add(GRU(300))</pre>
<pre>Layer (type) Output Shape</pre>	Enyer (type) Output Shape Param # embedding_2 (Embedding) (None, 2400, 300) 175133100 spatial_dropoutId_2 (Spatia (None, 2400, 300) 0 IDropoutID) gru_2 (GRU) (None, 300) 541800 dense_2 (Dense) (None, 1) 301 Potal params: 175,675,201 Trainable params: 542,101 Non-trainable params: 175,133,100 Mall time: 2.36 s model_fit(x_train_pad,ytrain, epochs = 1,batch_size = 128*strategy.num_replicas_in_sync) 362/1398 [=====>		
embedding_2 (Embedding) (None, 2400, 300) 175133100 spatial_dropoutld_2 (Spatia (None, 2400, 300) 0 1DropoutlD) gru_2 (GRU) (None, 300) 541800 dense_2 (Dense) (None, 1) 301 Total params: 175,675,201 Trainable params: 542,101 Non-trainable params: 175,133,100 Wall time: 2.36 s model_fit(x_train_pad,ytrain, epochs = 1,batch_size = 128*strategy.num_replicas_in_sync) 362/1398 [======>] - ETA: 30:58:28 - loss: 0.3277 - accuracy: 0.9044 gru_pred = model.predict(x_valid_pad) model_accuracy = roc_auc_score(yvalid,gru_pred) model_accuracy_ls.append({'model':'GRU','AUC_SCORE':model_accuracy})	<pre>embedding_2 (Embedding) (None, 2400, 300) 175133100 spatial_dropout1d_2 (Spatia (None, 2400, 300) 0 1Dropout1D) gru_2 (GRU) (None, 300) 541800 dense_2 (Dense) (None, 1) 301 Total params: 175,675,201 Trainable params: 542,101 Non-trainable params: 175,133,100 Wall time: 2.36 s model_fit(x_train_pad,ytrain, epochs = 1,batch_size = 128*strategy.num_replicas_in_sync) 362/1398 [======>] - ETA: 30:58:28 - loss: 0.3277 - accuracy: 0.9044 gru_pred = model_predict(x_valid_pad) model_accuracy = roc_auc_score(yvalid,gru_pred) model_accuracy_ls.append({'model':'GRU','AUC_SCORE':model_accuracy})</pre>	_	Layer (type) Output Shape Param #
<pre>IDropoutID) gru_2 (GRU)</pre>	IDropoutID) gru_2 (GRU) (None, 300) 541800 dense_2 (Dense) (None, 1) 301 Total params: 175,675,201 Trainable params: 542,101 Non-trainable params: 175,133,100 Wall time: 2.36 s model_fit(x_train_pad,ytrain, epochs = 1,batch_size = 128*strategy.num_replicas_in_sync) 362/1398 [=====>] - ETA: 30:58:28 - loss: 0.3277 - accuracy: 0.9044 gru_pred = model_predict(x_valid_pad) model_accuracy = roc_auc_score(yvalid,gru_pred) model_accuracy_ls.append({'model':'GRU','AUC_SCORE':model_accuracy})	=	embedding_2 (Embedding) (None, 2400, 300) 175133100
<pre>dense_2 (Dense)</pre>	<pre>dense_2 (Dense)</pre>		lDropout1D)
Total params: 175,675,201 Trainable params: 542,101 Non-trainable params: 175,133,100 Wall time: 2.36 s model.fit(x_train_pad,ytrain, epochs = 1,batch_size = 128*strategy.num_replicas_in_sync) 362/1398 [======>] - ETA: 30:58:28 - loss: 0.3277 - accuracy: 0.9044 gru_pred = model.predict(x_valid_pad) model_accuracy = roc_auc_score(yvalid,gru_pred) model_accuracy_ls.append({'model':'GRU','AUC_SCORE':model_accuracy})	Total params: 175,675,201 Prainable params: 542,101 Non-trainable params: 175,133,100 Wall time: 2.36 s model.fit(x_train_pad,ytrain, epochs = 1,batch_size = 128*strategy.num_replicas_in_sync) 362/1398 [=====>] - ETA: 30:58:28 - loss: 0.3277 - accuracy: 0.9044 gru_pred = model.predict(x_valid_pad) model_accuracy = roc_auc_score(yvalid,gru_pred) model_accuracy_ls.append({'model':'GRU','AUC_SCORE':model_accuracy})		
<pre>model.fit(x_train_pad,ytrain, epochs = 1,batch_size = 128*strategy.num_replicas_in_sync) 362/1398 [=====>] - ETA: 30:58:28 - loss: 0.3277 - accuracy: 0.9044 gru_pred = model.predict(x_valid_pad) model_accuracy = roc_auc_score(yvalid,gru_pred) model_accuracy_ls.append({'model':'GRU','AUC_SCORE':model_accuracy})</pre>	<pre>model.fit(x_train_pad,ytrain, epochs = 1,batch_size = 128*strategy.num_replicas_in_sync) 362/1398 [=====>] - ETA: 30:58:28 - loss: 0.3277 - accuracy: 0.9044 gru_pred = model.predict(x_valid_pad) model_accuracy = roc_auc_score(yvalid,gru_pred) model_accuracy_ls.append({'model':'GRU','AUC_SCORE':model_accuracy})</pre>	T N	Cotal params: 175,675,201 Crainable params: 542,101 Non-trainable params: 175,133,100
<pre>362/1398 [=====>] - ETA: 30:58:28 - loss: 0.3277 - accuracy: 0.9044 gru_pred = model.predict(x_valid_pad) model_accuracy = roc_auc_score(yvalid,gru_pred) model_accuracy_ls.append({'model':'GRU','AUC_SCORE':model_accuracy})</pre>	<pre>362/1398 [=====>] - ETA: 30:58:28 - loss: 0.3277 - accuracy: 0.9044 gru_pred = model.predict(x_valid_pad) model_accuracy = roc_auc_score(yvalid,gru_pred) model_accuracy_ls.append({'model':'GRU','AUC_SCORE':model_accuracy})</pre>		
<pre>model_accuracy = roc_auc_score(yvalid,gru_pred) model_accuracy_ls.append({'model':'GRU','AUC_SCORE':model_accuracy})</pre>	<pre>model_accuracy = roc_auc_score(yvalid,gru_pred) model_accuracy_ls.append({'model':'GRU','AUC_SCORE':model_accuracy})</pre>		
<pre>model_accuracy_ls.append({'model':'GRU','AUC_SCORE':model_accuracy})</pre>	<pre>model_accuracy_ls.append({'model':'GRU','AUC_SCORE':model_accuracy})</pre>		<pre>gru_pred = model.predict(x_valid_pad)</pre>
	model_accuracy_ls		

In [42]:

import numpy as np
import pandas as pd