User Response Classification Challenge

Adrien Cogny
Cornell Tech
ac2753@cornell.edu

April 23, 2018

Abstract

Abstract Text

I. Introduction

Chatbots are used in many applications today: customer support, flight booking, scheduling meeting, ordering food and many more. The application of a chatbot explored in this dataset is for a therapy chatbot. These types of chatbot, while very effective, may require human intervention. Determining when a human should intervene can be quite important, in this case when a person requires help in dealing with a complex situation, and requires tools to identify these situations.

The data set contains 80 examples of responses entered into a therapy chatbot. Each of these responses contains an id as well as an identification. The identification is either "flagged" if the response was flagged for human intervention or "not flagged" if not.

The task at hand was to create an AI agent to classify the user response.

II. Tools

The following tools and modules were used to complete this task:

- python 3.6.5 (using conda)
- pandas (0.22.0)
- dy-net (for the RNN) (2.0)
- scikit learn (for the Random Forest) (0.19.1)
- numpy (1.14.2)
- tqdm (for progress bars) (4.22.0)
- csv (For reading the csv embeddings to pandas) (1.0)
- re (regular expressions) (for cleaning the data) (2.2.1)

III. Preparing Data

The input data being sentences had to be cleaned up before passing into the models.

The first step was to load the csv file into a pandas dataframe and see what the data looked like. The data was, as mentioned above, a label as well as a sequence of words (not an array implementation yet). Due to the inherent nature of natural language processing both the label and sequence of words had to be converted to something which the machine could understand. That is, the label had to be converted from "flagged" or "not flagged" to 1 or 0 respectively and the sequence had to be converted to a series of word embeddings where each embedding represented a single word.

Natural language contains many words that are very common in sentences. These so-called stop words ("their", "he", "she", etc...) can make classifying a sentence very hard as, when combining word embeddings, they will take over the representation of the sentence simply by shear number. That is why in the preprocessing of each sentence (before it is given to the model), the stop words were removed from the sentence. By removing the stop words, we do not loose much important information and are able to classify more easily.

The models were created to do the conversion from label to 1 or 0 and from sentence of words to sequence of embeddings. The embeddings used were the GloVe 6B embeddings which come from wikipedia scapping[1].

IV. Models

i. RNN

When looking at sentence classification, one of the first thought was too look at an RNN encoder that would encode the sentence word by word and the computing a probability of being "flagged" or "not flagged". The label with the highest probability would then be applied to the sentence input.

i.1 RNN Description

DESCRIPTION OF RNN AND IMAGE/SKETCH OF THE MODEL CRATED

i.2 Tuning Parameters

When the model was created, the different parameters were tuned:

- Embedding Dimmension
- Hidden Dimmension Size
- Number of Epochs Run

The results for all of these tuning experiments are shown in the results section.

ii. Random Forest

After getting results for the RNN encoder and finding the best possible RNN, it was posited (based on research into text classification) that a random forest classifier could be more apt at this task.

[1] https://nlp.stanford.edu/projects/glove/

ii.1 Random Forest Description

RANDOM FOREST DESCRIPTION

Tuning Parameters When the model was created, the different parameters were tuned:

- Number of Estimators
- Sentence to Embedding Methodology

Whereas the number of estimators is a property of the random forest model itself, the sentence to embedding methodology describes how a sentences (or rather sequence of words) is transformed into a single vector wich can be input to the Random forest model.

There were two methods for embedding the sentence. One was to compute the mean of all the word embeddings and the other to compute the sum. The mean would attempt to construct a mean representation of the sentence using all the words in the sentence. The summation would create a sentence which was a sum of its parts.

V. Results

i. RNN

i.1 Hidden dimmension

The hidden dimmension test was done by keeping all parameters of the RNN constant except for the hidden dimmension of the RNN. The following figure shows the results for the hidden dimmension test performed. The test was performed by changing the size of the embedding dimmension from 0 to 9 dimmensions in steps of 1. The training loss, dev set true positive , dev set true negative , dev set false positive and dev set false negative were computed for each of the models run and a graph was created showing the true positive rate and the false positive rate.

	type	test	filename	maxEpochs	num_layers	embeddingSize	hiddenDim	train_loss	accuracy	truePos	trueNeg	falsePos	falseNeg
0	RNN	hidden_dimm	hidden_dimm_stats0.csv	400	1	50	0	36.1499	0.716667	0	17.2	6.8	0
1	RNN	hidden_dimm	hidden_dimm_stats1.csv	400	1	50	1	5.5576	0.579167	2.2	11.7	5.5	4.6
2	RNN	hidden_dimm	hidden_dimm_stats2.csv	400	1	50	2	0.886681	0.504167	2.6	9.5	5.7	6.2
3	RNN	hidden_dimm	hidden_dimm_stats3.csv	400	1	50	3	0.00841726	0.591667	2.4	11.8	5.3	4.5
4	RNN	hidden_dimm	hidden_dimm_stats4.csv	400	1	50	4	0.00756414	0.483333	2	9.6	5.1	7.3
5	RNN	hidden_dimm	hidden_dimm_stats5.csv	400	1	50	5	0.00441678	0.529167	1.9	10.8	4.9	6.4
6	RNN	hidden_dimm	hidden_dimm_stats6.csv	400	1	50	6	0.00633066	0.5375	1.6	11.3	4.6	6.5
7	RNN	hidden_dimm	hidden_dimm_stats7.csv	400	1	50	7	0.00421989	0.558333	1.9	11.5	5.8	4.8
8	RNN	hidden_dimm	hidden_dimm_stats8.csv	400	1	50	8	0.00384365	0.5125	1.9	10.4	6.1	5.6
9	RNN	hidden_dimm	hidden_dimm_stats9.csv	400	1	50	9	0.0032208	0.491667	2.3	9.5	5.2	7

Figure 1: *Shows the table of hidden dimmension tests for the RNN*

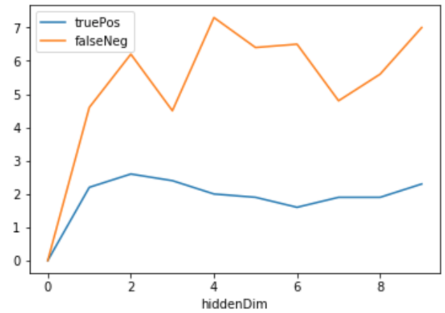


Figure 2: *Shows the graph of true positive count vs hidden dimmension of RNN*

This graph shows that the true positive count is always lower than the false negative count. This means that there are more instances where the model will classify a sentence as "not flagged" when in fact it should be "flagged" than there are instances where the model correctly classifies a sentence as "flagged." While this points towards this particular model (with the hyperparameters described below) not being good, the true positive rate and accuracies are derived (and much more important) metrics to look at.

While this graphs shows the true positive count as well as the false negative counts, a more interesting metric which can be derived fromt he true positive count and the false negative count is the true positive rate (tpr) which shows how much of the truth the model captures.

The following figure shows the tpr for the RNN model for different hidden dimmensions.

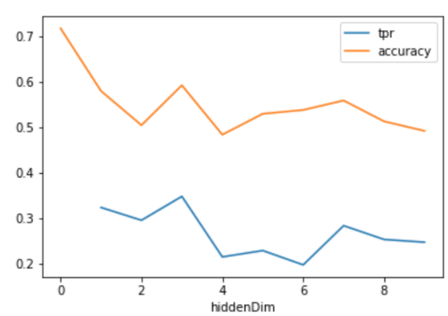


Figure 3: *Shows the graph of true positive rate vs hidden dimmension of RNN*

From this figure, we can clearly see that the tpr is greatest when the hidden dimmension is 3, with the other parameters set to: number of epochs ran = 400, number of layers = 1 and embedding size = 50. This model received an accuracy of 0.591667 and a true positive rate of 0.347826.

While the graph of accuracies shows that when the hidden dimension is of size 0, the accuracy jumps to 0.7, this model would not be considered to be a good model as the true positive rate is non existant because we are not classifying any results as being "flagged", which defeats the whole purpose of the model.

Thus, the hidden dimmension will be set to 3 for the other models.

i.2 GloVe embedding size

When the hidden dimmension was found, the next hyper-parameter which could be tuned was the size of the GloVe embeddings used to conver the sentences into something the models could understand.

This next figure shows the true positive counts and false negative counts for each embedding dimmension available (50, 100, 200 and 300).

type	test	filename	maxEpochs	num_layers	embeddingSize	hiddenDim	train_loss	accuracy	truePos	trueNeg	falsePos	falseNeg
RNN	embedding_dimm	embedd_size_stats50.csv	400	1	50	3	0.0081134	0.566667	2	11.6	5.26667	5.13333
RNN	embedding_dimm	embedd_size_stats100.csv	400	1	100	3	0.00727635	0.494444	2.26667	9.6	5.26667	6.86667
RNN	embedding_dimm	embedd_size_stats200.csv	400	1	200	3	0.00961309	0.591667	2.86667	11.3333	5.06667	4.73333
RNN	embedding_dimm	embedd_size_stats300.csv	400	1	300	3	0.00750879	0.505556	2.4	9.73333	5.4	6.46667

Figure 4: *Shows the table of embedding dimmension tests for the RNN*

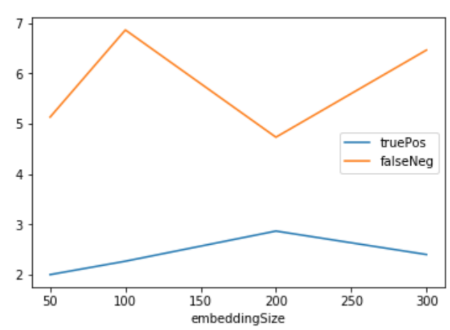


Figure 5: *Shows the graph of true positive count and false negative count vs embedding dimmensions of RNN*

The table and graph show that the true positive count is always smaller than the false negative count. However a dip can be seen in the false negative count at an embedding size of 200. This dip in false negatives is coupled by a rise in the true positive. This indicates that the embedding size of 200 yields the best results. [2]

Confirmation of this is seen in the figures below as both the true positive rate and accuracy are highest at an embedding size of 200.

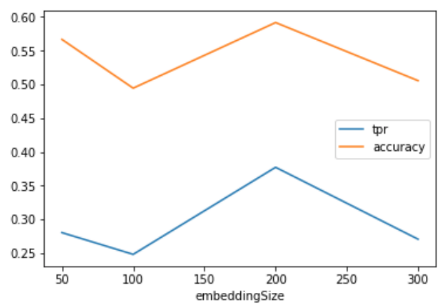
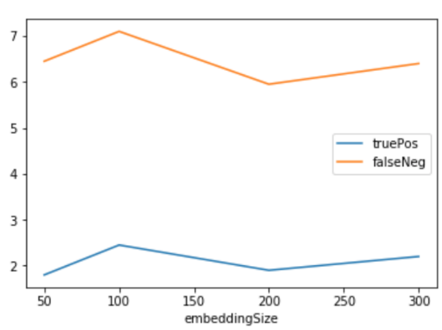
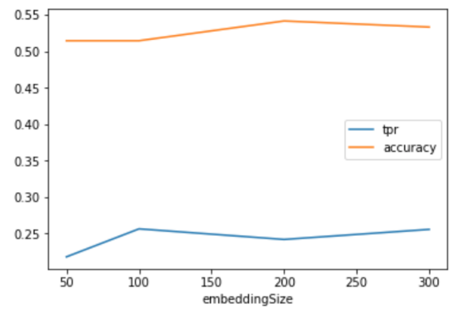


Figure 6: *Shows the graph of true positive rate vs embedding dimmensions of RNN*

However uppon further investigation, it seems that this may have been a random occurance due to the randomization of the data. After running this test several times, the graph did not show any significant improvement. For example, the following two figures was another run which illustrates that, depending on the random set of data, the best embeddings chnage



[2]Note that the increase in the true positive count and decrease in false negative counts graphed are not the same magnitude as the samples were drawn randomly from the training set and thus may not always contain the same number of positive and negative examples.



Therefore, for computation purposes, the embedding size was set to 100. This was thought to be a good compromise between a larger embedding which may provide more information and the speed of loading the embeddings.

i.3 Number Layers

The final hyperparameter to be tuned was the number of layers that the rnn contained. Before this test, the RNNs trained had a single layer. This test was performed twice in order to make sure there was consistency among even more random samples. The number of layers of the rnn were varied from 1 to 25 in increments of 1 for both tests.

The first test's graph shows the number of layers vs true positive count and false negative count is shown below:

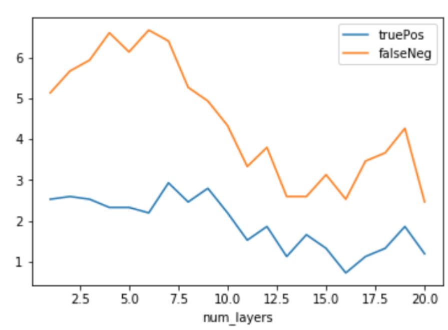


Figure 7: *Shows the graph of true positive rate vs number of layers of RNN for the first trial*

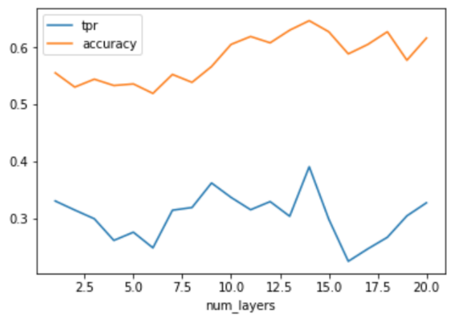


Figure 8: *Shows the graph of true positive rate vs number of layers of RNN for the first trial*

The second test's graph shows the same statistics as the previous one but for the second trial

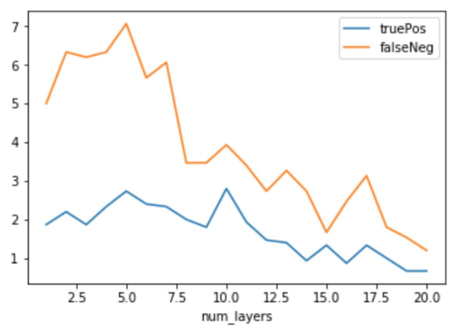


Figure 9: *Shows the graph of true positive rate vs number of layers of RNN for the second trial*

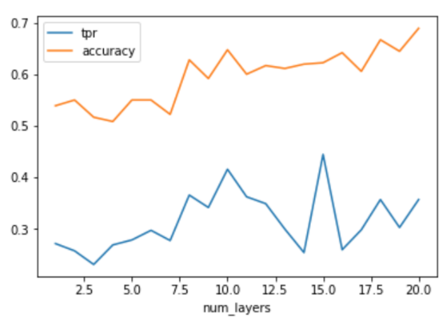


Figure 10: *Shows the graph of true positive rate vs number of layers of RNN for the second trial*

Both of these tests show that as the number of layers in the rnn increases the number of false negatives also increases. While the exact number of layers cannot be tuned very accuractely due to the fairly small dataset, it can be said that around 15 layers seems to be a fairly good point. That is because both the accuracy and the true positive rates are

i.4 RNN Final Model

The final RNN model had the following parameters:

- Hidden Dimmension = 3
- Number of Hidden Layers = 15
- Embedding Size = 100
- Embedding File = 'glove/glove.6B.100d.txt'
- Number of Epochs trained for = 400

The model was run 5 times to find the best true positive rate (TPR) on the dev set.

ii. Random Forest

As mentioned above, when the best rnn model was tuned it yielded an accuracy of

ii.1 Summation Methodology

	type	test	filename	n_estimators	accuracy	truePos	trueNeg	falsePos	falseNeg	tpr	prec
0	RandomForest	sum	randomFores_1_estimators	1	0.666667	3.8	12.2	3.86667	4.13333	0.478992	0.495652
1	RandomForest	sum	randomFores_2_estimators	2	0.672222	5.8	10.3333	2.73333	5.13333	0.530488	0.679688
2	RandomForest	sum	randomFores_3_estimators	3	0.733333	3.53333	14.0667	3.33333	3.06667	0.535354	0.514563
3	RandomForest	sum	randomFores_4_estimators	4	0.705556	4.93333	12	2.66667	4.4	0.528571	0.649123
4	RandomForest	sum	randomFores_5_estimators	5	0.730556	3.4	14.1333	3.86667	2.6	0.566667	0.46789
5	RandomForest	sum	randomFores_6_estimators	6	0.680556	4.73333	11.6	2.73333	4.93333	0.489655	0.633929
6	RandomForest	sum	randomFores_7_estimators	7	0.780556	3.06667	15.6667	3.2	2.06667	0.597403	0.489362
7	RandomForest	sum	randomFores_8_estimators	8	0.719444	3.13333	14.1333	3.86667	2.86667	0.522222	0.447619
8	RandomForest	sum	randomFores_9_estimators	9	0.736111	3.26667	14.4	3.93333	2.4	0.576471	0.453704
9	RandomForest	sum	randomFores_10_estimators	10	0.811111	4.33333	15.1333	2.53333	2	0.684211	0.631068
10	RandomForest	sum	randomFores_11_estimators	11	0.722222	3	14.3333	4.6	2.06667	0.592105	0.394737
11	RandomForest	sum	randomFores_12_estimators	12	0.741667	3.66667	14.1333	4	2.2	0.625	0.478261
12	RandomForest	sum	randomFores_13_estimators	13	0.747222	3.06667	14.8667	4.53333	1.53333	0.666667	0.403509
13	RandomForest	sum	randomFores_14_estimators	14	0.738889	3.66667	14.0667	4.2	2.06667	0.639535	0.466102
14	RandomForest	sum	randomFores_15_estimators	15	0.761111	3.6	14.6667	4.26667	1.46667	0.710526	0.457627
15	RandomForest	sum	randomFores_16_estimators	16	0.725	3.4	14	4.13333	2.46667	0.579545	0.451327
16	RandomForest	sum	randomFores_17_estimators	17	0.772222	3.6	14.9333	3.8	1.66667	0.683544	0.486486
17	RandomForest	sum	randomFores_18_estimators	18	0.727778	3.33333	14.1333	4.06667	2.46667	0.574713	0.45045
18	RandomForest	sum	randomFores_19_estimators	19	0.772222	3.8	14.7333	3.26667	2.2	0.633333	0.537736
19	RandomForest	sum	randomFores_20_estimators	20	0.744444	3.6	14.2667	3.73333	2.4	0.6	0.490909

Figure 11: *Shows the table for the experiments where the number of estimators used in the random forest classifier were used. This is for the summation methodology.*

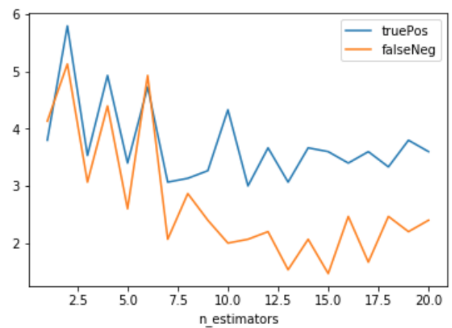


Figure 12: *Shows the graph for the experiments where the number of estimators used in the random forest classifier were used. This is for the summation methodology.*

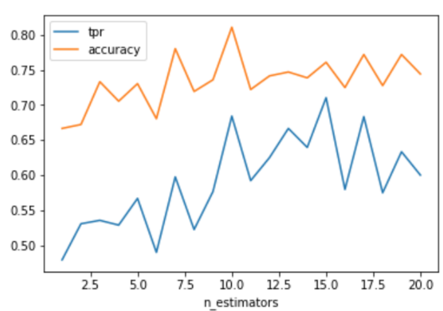


Figure 13: *Shows the graph of true positive rate and accuracy vs number of estimators of random forest with summed embeddings as representation for the sentence*

ii.2 Mean Methodology

	type	test	filename	n_estimators	accuracy	truePos	trueNeg	falsePos	falseNeg	tpr	prec
20	RandomForest	mean	randomForest_meanNotSum_1_estimators	1	0.55	2.86667	10.3333	6.2	4.6	0.383929	0.316176
21	RandomForest	mean	randomForest_meanNotSum_2_estimators	2	0.586111	5.46667	8.6	3.4	6.53333	0.455556	0.616541
22	RandomForest	mean	randomForest_meanNotSum_3_estimators	3	0.655556	2.73333	13	4.66667	3.6	0.431579	0.369369
23	RandomForest	mean	randomForest_meanNotSum_4_estimators	4	0.644444	4.06667	11.4	4.06667	4.46667	0.476562	0.5
24	RandomForest	mean	randomForest_meanNotSum_5_estimators	5	0.658333	3	12.8	5.2	3	0.5	0.365854
25	RandomForest	mean	randomForest_meanNotSum_6_estimators	6	0.661111	4.06667	11.8	3.33333	4.8	0.458647	0.54955
26	RandomForest	mean	randomForest_meanNotSum_7_estimators	7	0.675	3.13333	13.0667	4.73333	3.06667	0.505376	0.398305
27	RandomForest	mean	randomForest_meanNotSum_8_estimators	8	0.652778	3.66667	12	4	4.33333	0.458333	0.478261
28	RandomForest	mean	randomForest_meanNotSum_9_estimators	9	0.683333	2.06667	14.3333	4.86667	2.73333	0.430556	0.298077
29	RandomForest	mean	randomForest_meanNotSum_10_estimators	10	0.675	2.53333	13.6667	4.93333	2.86667	0.469136	0.339286
30	RandomForest	mean	randomForest_meanNotSum_11_estimators	11	0.705556	3.13333	13.8	4.4	2.66667	0.54023	0.415929
31	RandomForest	mean	randomForest_meanNotSum_12_estimators	12	0.658333	2.4	13.4	4.4	3.8	0.387097	0.352941
32	RandomForest	mean	randomForest_meanNotSum_13_estimators	13	0.688889	3.26667	13.2667	5.46667	2	0.620253	0.374046
33	RandomForest	mean	randomForest_meanNotSum_14_estimators	14	0.65	2.4	13.2	5.06667	3.33333	0.418605	0.321429
34	RandomForest	mean	randomForest_meanNotSum_15_estimators	15	0.730556	3.13333	14.4	4.4	2.06667	0.602564	0.415929
35	RandomForest	mean	randomForest_meanNotSum_16_estimators	16	0.702778	3.93333	12.9333	4.33333	2.8	0.584158	0.475806
36	RandomForest	mean	randomForest_meanNotSum_17_estimators	17	0.686111	2.93333	13.5333	4.26667	3.26667	0.473118	0.407407
37	RandomForest	mean	randomForest_meanNotSum_18_estimators	18	0.691667	2.93333	13.6667	4.8	2.6	0.53012	0.37931
38	RandomForest	mean	randomForest_meanNotSum_19_estimators	19	0.711111	3.06667	14	4.53333	2.4	0.560976	0.403509
39	RandomForest	mean	randomForest_meanNotSum_20_estimators	20	0.711111	3	14.0667	4.46667	2.46667	0.54878	0.401786

Figure 14: *Shows the table for the experiments where the number of estimators used in the random forest classifier were used. This is for the mean methodology.*

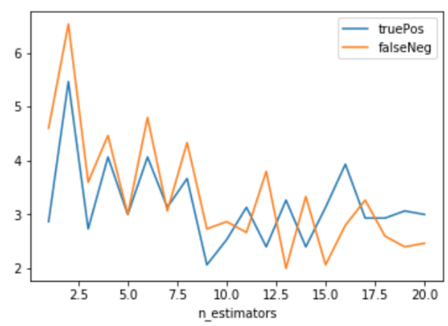


Figure 15: *Shows the graph for the experiments where the number of estimators used in the random forest classifier were used. This is for the summation methodology.*

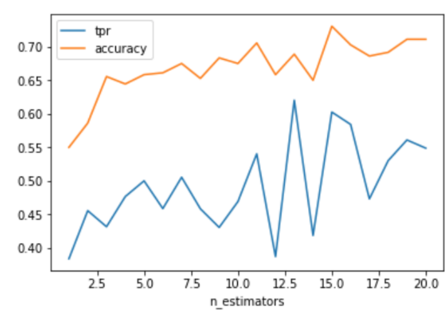


Figure 16: *Shows the graph of true positive rate and accuracy vs number of estimators of random forest with summed embeddings as representation for the sentence*

ii.3 Mean vs Summation Methodology

From the figures above, it can be seen that, for all numbers of estimators used in the random forest mode the summation accuracy and true positive rates are higher than the mean methodology. This makes sense as the summation method attempts to feed the model the summation of the words rather than the average representation.

ii.4 Random Forest Final Model

The final model for the random forest had the following paramters:

iii. Final Model

iv. Using the Model Explanation of Code

References

[Figueredo and Wolf, 2009] Figueredo, A. J. and Wolf, P. S. A. (2009). Assortative pairing and life history strategy - a cross-cultural study. *Human Nature*, 20:317–330.