Machine Learning Engineer Nanodegree

Capstone Project

Andres Rubio del Saz

May 19th, 2021

Definition

1. Project Overview

When I started this nanodegree one of the things I wanted to study was about NLP and sentiment analysis trying then to apply to different areas in my company because of that in this final capstone I have tried to create an sentiment analysis based on reviews and scoring of Amazon pet supplier dataset testing different approaches and capabilities of different AWS machines in order to understand computational times and billing. This dataset has the following structure:

	reviewText	overall	summary
0	I purchased the Trilogy with hoping my two cat...	3	Nice Distraction for my cats for about 15 minutes
1	There are usually one or more of my cats watch...	5	Entertaining for my cats
2	I bought the triliogy and have tested out all ...	4	Entertaining
3	My female kitty could care less about these vi...	4	Happy to have them
4	If I had gotten just volume two, I would have ...	3	You really only need vol 2

ReviewText with the long review which in the end I did not use and I focused in the summary one
Summary has a small review that concentrates all the feeling of the customer that is why I use this one for creating vocabulary and train the model
Overall is the score of the customer to the review and is the output that I try to predict

Taking this into account I will try to create a model based on RNN to predict the overall, and compare performance between 2 ways of creating RNN with LSTM and GRU types.

Problem Statement

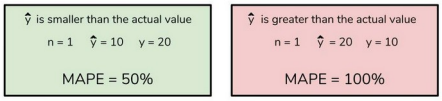
To create a sentiment analysis with Deep Learning according to this nanodegree a good option is a RNN using Pytorch to create neural nets that are able to predict how good a text is according the feeling of the customer and the words that uses.

So in this capstone I have implemented a RNN architecture based on LSTM and GRU to predict this sentiment according to the score (overall) which will be the output of our net. Although the overall is a value between 1-5 I have normalized this number to be a number between 0-1 to work better with the functions that Neural networks have using Pytorch and Sagemaker.

Metrics

To test if we are doing good predictions with out training and in the evaluation of the model I have used SMAPE (Symetric Mean Absolute Percentage Error)and MAPE (Mean absolute percentage error) which are commonly used as model evaluation although others can be found in the notebook.

$$MAPE = \frac{100\%}{n} \Sigma \left| \frac{y - \widehat{y}}{y} \right|$$



$$\text{SMAPE} = \frac{1}{n} \sum_{t=1}^{n} \frac{|F_t - A_t|}{(|A_t| + |F_t|)/2}$$

I did not get good results when I evaluate the model which is a thing I would like to understand better in this review in the final section I have explained my thoughts about that. Another thing to take into account is that in my proposal I chose accuracy metric that took into account true positives and true negatives divided by data size which I realized that it was a bad metric to test model accuracy because output of the net is not 1 or 0 but a value between 0 and 1

2. Analysis

Data Exploration

As was told in the capstone project, data about suppliers has been downloaded from http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/reviews_Pet_Supplies_5.json.gz in local and loaded inside a Pandas dataframe deleting the fields that we consider we do not need to create our model and the vocabulary. The final picture is the following

	reviewText	overall	summary
0	I purchased the Trilogy with hoping my two cat...	3	Nice Distraction for my cats for about 15 minutes
1	There are usually one or more of my cats watch...	5	Entertaining for my cats
2	I bought the triliogy and have tested out all ...	4	Entertaining
3	My female kitty could care less about these vi...	4	Happy to have them
4	If I had gotten just volume two, I would have ...	3	You really only need vol 2

The final number or records in this stage is ***157836***

Algorithms and Techniques

We have tried to create models of RNN an LSTM and GRU to compare performance.

The LSTM architecture has been the following

```
LSTMClassifier(
  (embedding): Embedding(7264, 400, padding_idx=0)
  (lstm): LSTM(400, 256, num_layers=2, batch_first=True, dropout=0.5)
  (dropout): Dropout(p=0.5, inplace=False)
  (dense): Linear(in_features=256, out_features=1, bias=True)
  (activation): LeakyReLU(negative_slope=0.01)
)
```

The GRU architecture has been the following

```
GRUNet(
  (embedding): Embedding(7264, 400, padding_idx=0)
  (gru): GRU(400, 256, num_layers=2, batch_first=True, dropout=0.5)
  (dropout): Dropout(p=0.5, inplace=False)
  (fc): Linear(in_features=256, out_features=1, bias=True)
  (relu): ReLU()
)
```

Loss function: **MSELoss** creates a criterion that measures the mean squared error (squared L2 norm) between each element in the input x and target y .

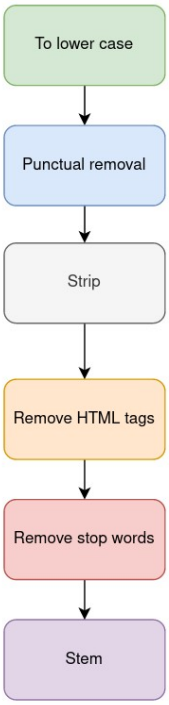
Learning Rate (LR) = 0.001 as a step size for iteration

I saw that GRU is faster but I did not get good results when I test error in evaluation

3. Methodology

Data Preprocessing

In the very beginning of the project I thought I would use the ***reviewText*** field to train the model and the vocabulary but I realized that with the ***summary*** field could be enough for that. So I apply some transformations to that field as following

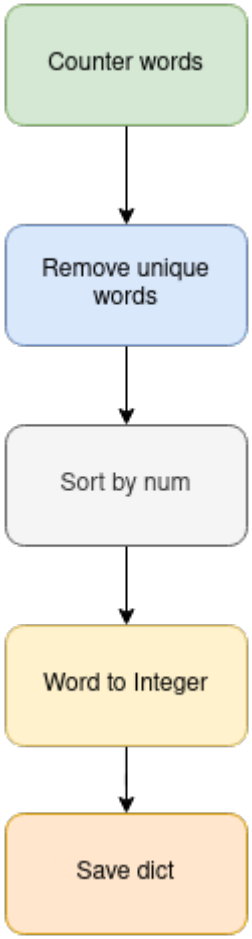


The final result in this stage is the following:

	reviewText	overall	summary	summary_	summary__
0	i purchased the trilogy with hoping my two cat...	3	nice distraction for my cats for about 15 minutes	nice distraction for my cats for about 15 minutes	[nice, distract, cat, 15, minut]
1	there are usually one or more of my cats watch...	5	entertaining for my cats	entertaining for my cats	[entertain, cat]
2	i bought the triliogy and have tested out all ...	4	entertaining	entertaining	[entertain]
3	my female kitty could care less about these vi...	4	happy to have them	happy to have them	[happi]
4	if i had gotten just volume two, i would have ...	3	you really only need vol 2	you really only need vol 2	[realli, need, vol, 2]

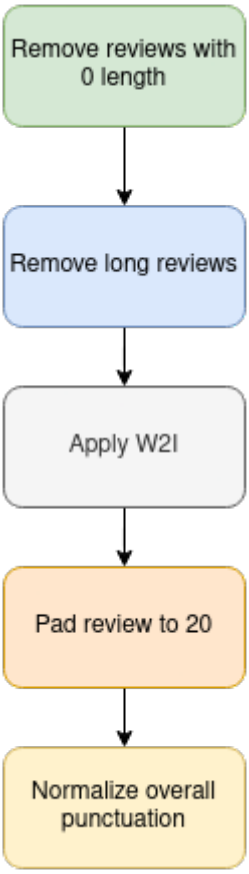
Implementation

Now I create a vocabulary based on the previous dataframe according the following flow



Now we have a vocabulary to apply for each review in order to translate words to integer to feed the net. *Save dict will not be useful in the end.

Now we apply this dictionary to the review to have a hot encoding vector according the following schema:



After applying W2I this is the final dataframe

	reviewText	overall	summary	summary_	summary__	W2I	overall_
0	i purchased the trilogy with hoping my two cat...	3	nice distraction for my cats for about 15 minutes	nice distraction for my cats for about 15 minutes	[nice, distract, cat, 15, minut]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.6
1	there are usually one or more of my cats watch...	5	entertaining for my cats	entertaining for my cats	[entertain, cat]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	1.0
2	i bought the triliogy and have tested out all ...	4	entertaining	entertaining	[entertain]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.8
3	my female kitty could care less about these vi...	4	happy to have them	happy to have them	[happi]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.8
4	if i had gotten just volume two, i would have ...	3	you really only need vol 2	you really only need vol 2	[realli, need, vol, 2]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.6
...	...	...	...	...	...	...	...
157831	the pet magasin retractable dog leash is the b...	5	perfection	perfection	[perfect]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	1.0
157832	i'm not the biggest fan of retractable leashes...	5	nice solid feel, my little boy is thrilled!	nice solid feel, my little boy is thrilled!	[nice, solid, feel, littl, boy, thrill]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 15,...	1.0
157833	i was provided a leash for my unbiased review....	4	good leash for small non-darty pets	good leash for small non-darty pets	[good, leash, small, non, darti, pet]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, ...	0.8
157834	this is a well designed leash that offers the ...	5	quality build, smooth retraction, great leash	quality build, smooth retraction, great leash	[qualiti, build, smooth, retract, great, leash]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 19,...	1.0
157835	this leash is stated to be good for dogs up to...	4	nice leash for 20+ pounds, occasional problems...	nice leash for 20+ pounds, occasional problems...	[nice, leash, 20, pound, occasion, problem]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 15,...	0.8

157531 rows × 7 columns

Once we have applied the dictionary to the dataframe creating the W2I field which contains the data that is going to be used to train the algorithm I took advantage of sklearn split with test_size=0.33, random_state=42 as parameters.

The result is the following for training data

```
26897     [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
149426    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
100956    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
45653     [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
55633     [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
                                ...
120112    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
103888    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
132193    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
147158    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
122196    [0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 133, 1588, 755,...
Name: W2I, Length: 105545, dtype: object
```

and the expected output review score as:

```
26897     1.0
149426    1.0
100956    1.0
45653     0.8
55633     0.8
          ...
120112    0.8
103888    1.0
132193    0.8
147158    1.0
122196    0.8
Name: overall_, Length: 105545, dtype: float64
```

It is remarkable that in the beginning of the project I thought I can use the *scale 1 – 5 as output* of the RNN but all the documentation says that better values between *0 – 1* that's is why I normalize the field

With that split we try to create tensors in pytorch, as an example

```
Sample input size:  torch.Size([50, 20])
Sample input:
 tensor([[   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,   11,  285,  525,   50],
        [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    6,  181, 4473,   48],
        [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    2,   90],
        [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,   15,   28,   67,  583],
        [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    7,    2,  216,   64],
        [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,  887,  595,  190,  805,  542,  334, 1496],
        [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,  650,    3,  997],
        [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    6,  123,   96,    4],
        [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,   21,  374,  276,   95,   21],
```

Refinement

I have faced lots of problems during training following the ones that made me suffer the most.

- I have had problems with tensor's shape because they did not match
- I have had problems understanding activation functions. I thought I could use Softmax with scale 1-5 but I did not get valid results
- I have had problems with loss functions understanding which one was suitable according activation function.
- I have had problems understanding how to evaluate the training because of data structures and playing with tensors to numpy arrays
- I have had problems with sagemaker and how binaries are stored in AWS machines. Although in the end I did not finish Sagemaker part because a problem that I had and I will explain in the final chapter.
- I have had problems with numpy binaries
- I have had problems with CUDA memory when allocate memory in evaluation phase

4. Results

Model Evaluation and Validation

LSTM 3 epochs, LR=0.001, 2 Fully connected layers, LeakyRelu as activation function,

```
LSTMClassifier(
  (embedding): Embedding(7264, 400, padding_idx=0)
  (lstm): LSTM(400, 256, num_layers=2, batch_first=True, dropout=0.5)
  (dropout): Dropout(p=0.5, inplace=False)
  (dense): Linear(in_features=256, out_features=1, bias=True)
  (activation): LeakyReLU(negative_slope=0.01)
)
```

```
Epoch 1......Step: 200/2111....... Average Loss for Epoch: 0.09881627066060901
Epoch 1......Step: 400/2111....... Average Loss for Epoch: 0.0797984169377014
Epoch 1......Step: 600/2111....... Average Loss for Epoch: 0.07329722442974647
Epoch 1......Step: 800/2111....... Average Loss for Epoch: 0.06934167568804696
Epoch 1......Step: 1000/2111....... Average Loss for Epoch: 0.06719747389107943
Epoch 1......Step: 1200/2111....... Average Loss for Epoch: 0.06552978387102484
Epoch 1......Step: 1400/2111....... Average Loss for Epoch: 0.06452119627435292
Epoch 1......Step: 1600/2111....... Average Loss for Epoch: 0.06371485159499571
Epoch 1......Step: 1800/2111....... Average Loss for Epoch: 0.06293680767839153
Epoch 1......Step: 2000/2111....... Average Loss for Epoch: 0.06252600147109479
```

```
Epoch 1/3 Done, Total Loss: 0.062177731407155 98
Total Time Elapsed: 37.859999999999985 seconds
Total Training Time: 37.859999999999985 seconds
Epoch 2......Step: 200/2111....... Average Loss for Epoch: 0.05673592926003039
Epoch 2......Step: 400/2111....... Average Loss for Epoch: 0.057097256216220554
Epoch 2......Step: 600/2111....... Average Loss for Epoch: 0.0575214586344858
Epoch 2......Step: 800/2111....... Average Loss for Epoch: 0.05722232834203169
Epoch 2......Step: 1000/2111....... Average Loss for Epoch: 0.05727312531881034
Epoch 2......Step: 1200/2111....... Average Loss for Epoch: 0.05708364626237502
Epoch 2......Step: 1400/2111....... Average Loss for Epoch: 0.05715849434158632
Epoch 2......Step: 1600/2111....... Average Loss for Epoch: 0.05715720699517988
Epoch 2......Step: 1800/2111....... Average Loss for Epoch: 0.05701097245535089
Epoch 2......Step: 2000/2111....... Average Loss for Epoch: 0.0571044690143317
Epoch 2/3 Done, Total Loss: 0.05699185032551038
Total Time Elapsed: 37.70000000000002 seconds
Total Training Time: 75.56 seconds
Epoch 3......Step: 200/2111....... Average Loss for Epoch: 0.05591134018264711
Epoch 3......Step: 400/2111....... Average Loss for Epoch: 0.05631893162149936
Epoch 3......Step: 600/2111....... Average Loss for Epoch: 0.05673748734717568
Epoch 3......Step: 800/2111....... Average Loss for Epoch: 0.056454298791941256
Epoch 3......Step: 1000/2111....... Average Loss for Epoch: 0.0565092316120863
Epoch 3......Step: 1200/2111....... Average Loss for Epoch: 0.056327881077304486
Epoch 3......Step: 1400/2111....... Average Loss for Epoch: 0.056424185899751525
Epoch 3......Step: 1600/2111....... Average Loss for Epoch: 0.056440231062006206
Epoch 3......Step: 1800/2111....... Average Loss for Epoch: 0.05631134949831499
Epoch 3......Step: 2000/2111....... Average Loss for Epoch: 0.05642174789775163
Epoch 3/3 Done, Total Loss: 0.05631042017470401
Total Time Elapsed: 37.77000000000001 seconds
Total Training Time: 113.33000000000001 seconds
Epoch: 3, Loss: 0.0
```

GRU model

```
GRUNet(
  (embedding): Embedding(7264, 400, padding_idx=0)
  (gru): GRU(400, 256, num_layers=2, batch_first=True, dropout=0.5)
  (dropout): Dropout(p=0.5, inplace=False)
  (fc): Linear(in_features=256, out_features=1, bias=True)
  (relu): ReLU()
)
```

```
Epoch 1......Step: 200/2111....... Average Loss for Epoch: 0.6266861668229103
Epoch 1......Step: 400/2111....... Average Loss for Epoch: 0.6282054242491723
Epoch 1......Step: 600/2111....... Average Loss for Epoch: 0.6270927435656388
Epoch 1......Step: 800/2111....... Average Loss for Epoch: 0.6273189376667142
Epoch 1......Step: 1000/2111....... Average Loss for Epoch: 0.6263362368941308
Epoch 1......Step: 1200/2111....... Average Loss for Epoch: 0.626381298204263
Epoch 1......Step: 1400/2111....... Average Loss for Epoch: 0.6258667463915689
Epoch 1......Step: 1600/2111....... Average Loss for Epoch: 0.6257400956004858
Epoch 1......Step: 1800/2111....... Average Loss for Epoch: 0.6260462725162506
Epoch 1......Step: 2000/2111....... Average Loss for Epoch: 0.62540500728786
```

```
Epoch 1/3 Done, Total Loss: 0.6252916760068548
Total Time Elapsed: 34.75 seconds
Total Training Time: 34.75 seconds
Epoch 2.....Step: 200/2111..... Average Loss for Epoch: 0.6269868022203445
Epoch 2.....Step: 400/2111..... Average Loss for Epoch: 0.6279549877345562
Epoch 2.....Step: 600/2111..... Average Loss for Epoch: 0.6256985849142075
Epoch 2.....Step: 800/2111..... Average Loss for Epoch: 0.6265788702666759
Epoch 2.....Step: 1000/2111..... Average Loss for Epoch: 0.6255598131418229
Epoch 2.....Step: 1200/2111..... Average Loss for Epoch: 0.6259095249076684
Epoch 2.....Step: 1400/2111..... Average Loss for Epoch: 0.6255333759954997
Epoch 2.....Step: 1600/2111..... Average Loss for Epoch: 0.6253826293721795
Epoch 2.....Step: 1800/2111..... Average Loss for Epoch: 0.6257050559255812
Epoch 2.....Step: 2000/2111..... Average Loss for Epoch: 0.6249732454121113
Epoch 2/3 Done, Total Loss: 0.6251425569738499
Total Time Elapsed: 34.72000000000003 seconds
Total Training Time: 69.47000000000003 seconds
Epoch 3.....Step: 200/2111..... Average Loss for Epoch: 0.6270450669527053
Epoch 3.....Step: 400/2111..... Average Loss for Epoch: 0.6268577967584134
Epoch 3.....Step: 600/2111..... Average Loss for Epoch: 0.6258833941817283
Epoch 3.....Step: 800/2111..... Average Loss for Epoch: 0.6262158544361591
Epoch 3.....Step: 1000/2111..... Average Loss for Epoch: 0.6251552116274833
Epoch 3.....Step: 1200/2111..... Average Loss for Epoch: 0.6256068044404188
Epoch 3.....Step: 1400/2111..... Average Loss for Epoch: 0.6247454785874912
Epoch 3.....Step: 1600/2111..... Average Loss for Epoch: 0.6247106900066137
Epoch 3.....Step: 1800/2111..... Average Loss for Epoch: 0.6249771651294497
Epoch 3.....Step: 2000/2111..... Average Loss for Epoch: 0.6243294233828783
Epoch 3/3 Done, Total Loss: 0.6242959179594753
Total Time Elapsed: 34.79000000000002 seconds
Total Training Time: 104.26000000000005 seconds
Epoch: 3, Loss: 0.0
```

For Evaluation model I did not get good results looking at the accuracy of _symmetrical_mean_absolute_percentage_error and _mean_absolute_percentage_error not sure why.

My assumptions in this regard is because the training is failing because broadcasting is not working properly because of the input data or because I am not using a proper loss function and activation function.

I am pretty sure that input data format is fine but I do not of my RNN architecture and parametrization of it. I would appreciate feedback about it

5. Conclusion

Reflection

I have tested with different AWS instances with GPU also watching how different the speed of the training was. Because of that I spent all the budget although I was very carefully destroying resources and stopping them.

When I was ready to finish and upload the project my setup was destroyed because of that and I had to redo everything again which was exhausted.

I could not reach all my goals because I had several problems that were incredible difficult like for example binaries in numpy to use form Sagemaker scripts. It was also challenging the different versions of Sagemaker because some material of the course is outdated in this regard and I did not get very thrilling to use Sagemaker in the end because I was very focus on using pytorch for the first time.

I would like to test different RNN architectures and maybe change vocab size maybe with a reduced amount of words.

I saw that 3 epocs was enough to get better result but I think that is because some problems in the training that I would like to understand. It is very difficult to progress in this regard only chating in the platform and a better support must be added specially for the cost of the course and the amount of work that the student has to do. Furthermore material is outdated.

Improvement

My final results are not very good and disappointing. I would appreciate some feedback of how I can improve the accuracy of the training and I can say in the end that I have learned a lot facing all the problems.
I think methodology with preprocessing data is correct and the normalize the overall field was a good choice. I think I need to improve the tensor's conditionality because for some reasons it looks like affect the broadcasting of the net and can be affect accuracy.

Maybe a reduced vocabulary can improve the performance because of the hidden dimensions of the layers and the embedding. No idea.

Not sure if a different activation function and loss function can improve results in some way I think it is the right choice and I think maybe it is the way I do input data.

I could not test all with the sagemaker endpoints and hyper tunning, I do not see the endpoint capability suitable for the real world there are different solutions not so AWS and maybe hyper tunning it is interesting.

Also Udacity needs to improve the course because it is an online course with no guidance very expensive when all the work has to be done by the student and face problems with the platform, budget and very long descriptions.