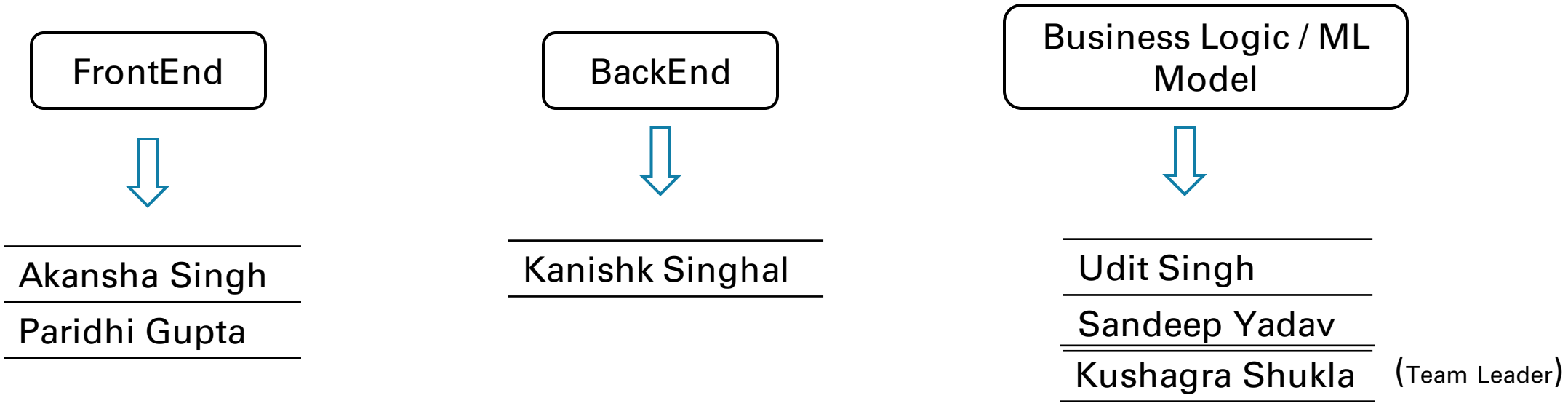
Code.Black()

NM396: SENTIMENT ANALYSIS AND CUSTOMER REVIEW RATING

Team Structure



Basic Functionality

INPUT TYPES

Single Review

Bulk Review (csv file)

Processing Input

OUTPUT TYPES

Rating of 1 to 5

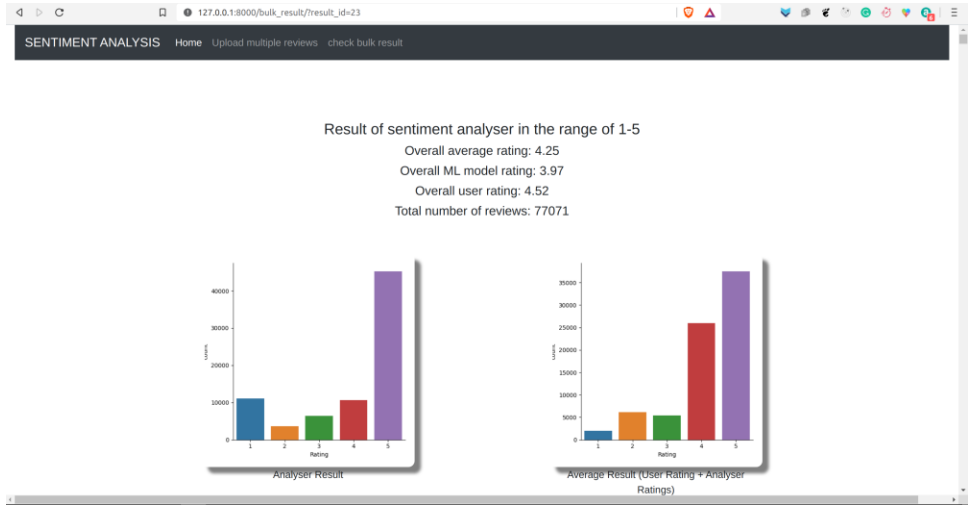
Tabular + Graphical O/P

User Rating, Review Text, Model Sentiment Rating , Avg. Product Rating

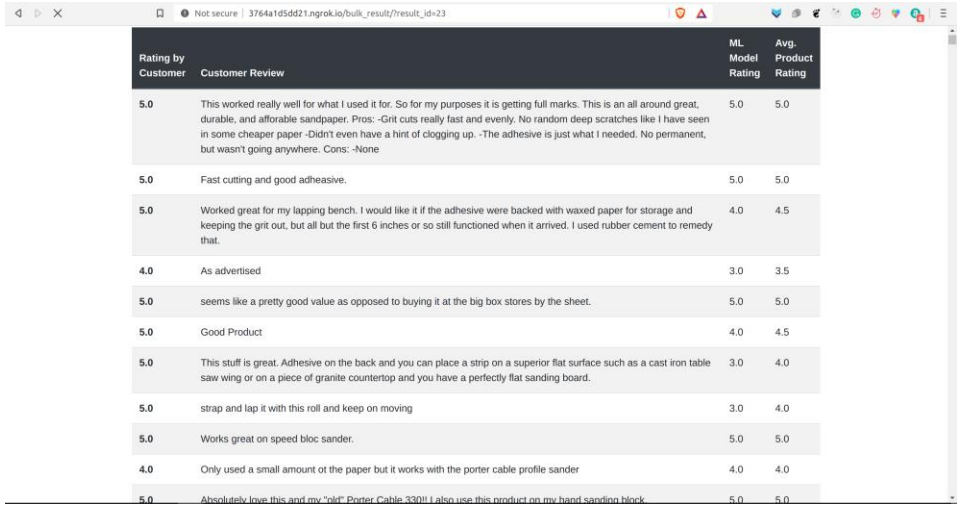
Note:
1 -> Worst (High Negative)
5 -> Best (High Positive)
Rating output by model on the scale of 1 to 5 represents **degree of the sentiment**

Input

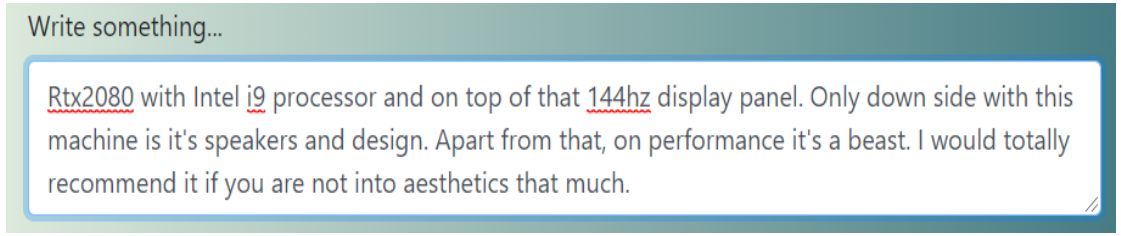
csv file (with 77k reviews



O/P



Rating by Customer	Customer Review	ML Model Rating	Avg. Product Rating
5.0	This worked really well for what I used it for. So for my purposes it is getting full marks. This is an all around great, durable, and afforable sandpaper. Pros: -Grit cuts really fast and evenly. No random deep scratches like I have seen in some cheaper paper -Didn't even have a hint of clogging up. -The adhesive is just what I needed. No permanent, but wasn't going anywhere. Cons: -None	5.0	5.0
5.0	Fast cutting and good adheasive.	5.0	5.0
5.0	Worked great for my lapping bench. I would like it if the adhesive were backed with waxed paper for storage and keeping the grit out, but all but the first 6 inches or so still functioned when it arrived. I used rubber cement to remedy that.	4.0	4.5
4.0	As advertised	3.0	3.5
5.0	seems like a pretty good value as opposed to buying it at the big box stores by the sheet.	5.0	5.0
5.0	Good Product	4.0	4.5
5.0	This stuff is great. Adhesive on the back and you can place a strip on a superior flat surface such as a cast iron table saw wing or on a piece of granite countertop and you have a perfectly flat sanding board.	3.0	4.0
5.0	strap and lap it with this roll and keep on moving	3.0	4.0
5.0	Works great on speed bloc sander.	5.0	5.0
4.0	Only used a small amount ot the paper but it works with the porter cable profile sander	4.0	4.0
5.0	Absolutely love this and my "old" Porter Cable 330!! I also use this product on my hand sanding block	5.0	5.0

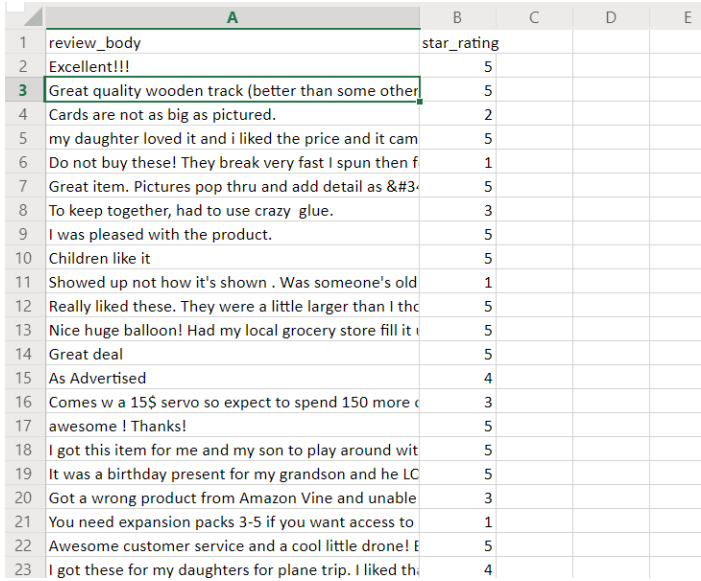


Result of sentiment analyser in the range of 1-5

Sentence provided: Rtx2080 with Intel i9 processor and on top of that 144hz display panel. Only down side with this machine is it's speakers and design. Apart from that, on performance it's a beast. I would totally recommend it if you are not into aesthetics that much.

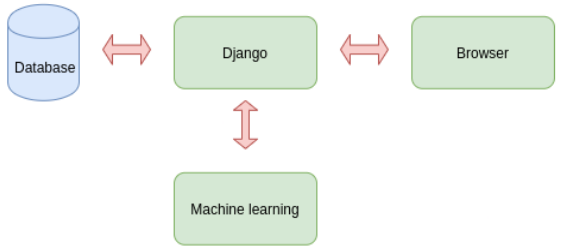
Overall average rating: [4]

Note:
csv file structure
Row 1 (headers) :
review_text , user_ratings
All other rows:
<text>, <digit in [1,5]>

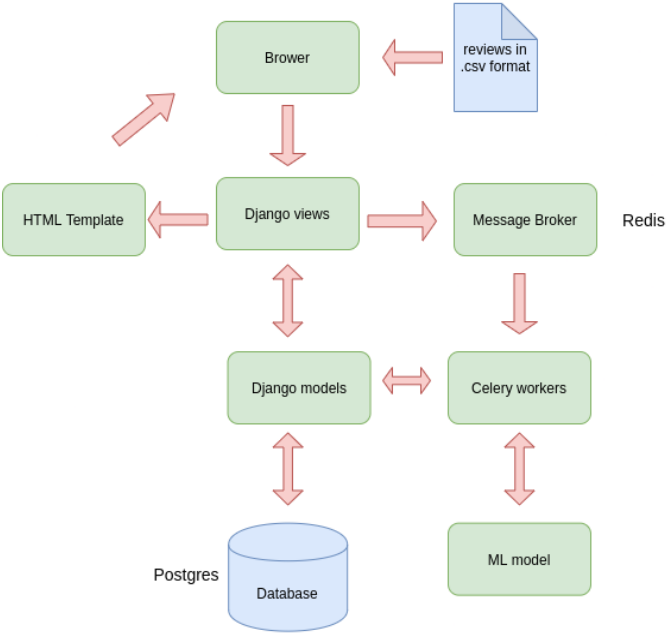


	A	B	C	D	E
1	review_body	star_rating			
2	Excellent!!!	5			
3	Great quality wooden track (better than some other	5			
4	Cards are not as big as pictured.	2			
5	my daughter loved it and i liked the price and it cam	5			
6	Do not buy these! They break very fast I spun then f	1			
7	Great item. Pictures pop thru and add detail as &#3	5			
8	To keep together, had to use crazy glue.	3			
9	I was pleased with the product.	5			
10	Children like it	5			
11	Showed up not how it's shown . Was someone's old	1			
12	Really liked these. They were a little larger than I thc	5			
13	Nice huge balloon! Had my local grocery store fill it	5			
14	Great deal	5			
15	As Advertised	4			
16	Comes w a 15$ servo so expect to spend 150 more	3			
17	awesome ! Thanks!	5			
18	I got this item for me and my son to play around wit	5			
19	It was a birthday present for my grandson and he LC	5			
20	Got a wrong product from Amazon Vine and unable	3			
21	You need expansion packs 3-5 if you want access to	1			
22	Awesome customer service and a cool little drone! E	5			
23	I got these for my daughters for plane trip. I liked th	4			

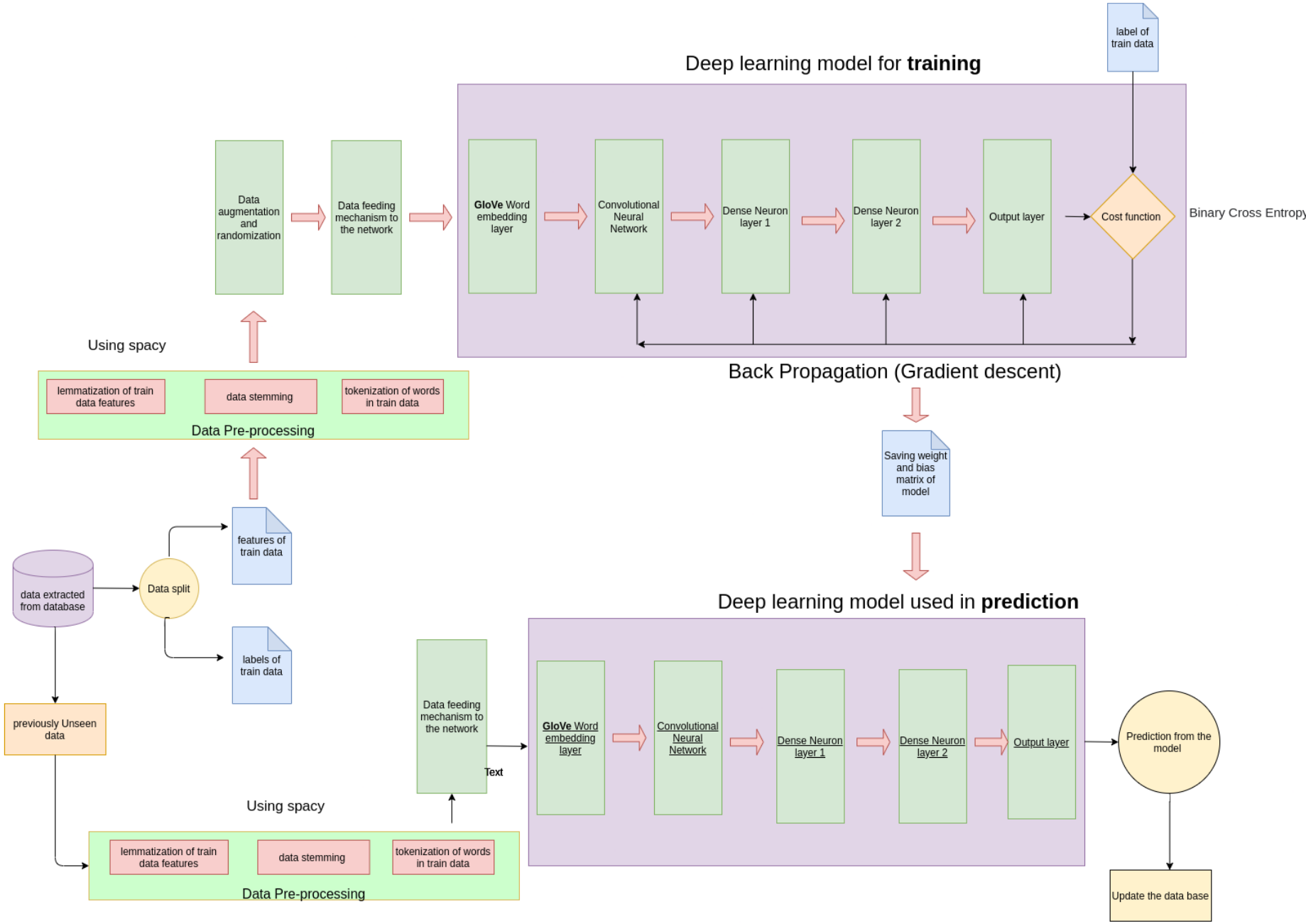
Under the Hood



High level view of system



Django Backend Architecture



Data flow through our ML model

We do things differently

We use CNN. Why?

- Less runtime after training
- Less weights to be trained
- Model size is relatively small for same accuracy
- Work on low power machines.

We use Job Queueing. How & Why?

- Celery for an **asynchronous** task queue based on distributed message passing
- Redis as blazing-fast **message broker**
- We need not to wait because somebody else uploaded a csv file with **1 lakh entries**.

Result ?

93 %

accuracy

+

We can handle multiple requests simultaneously

Model in the making

Training Metrics

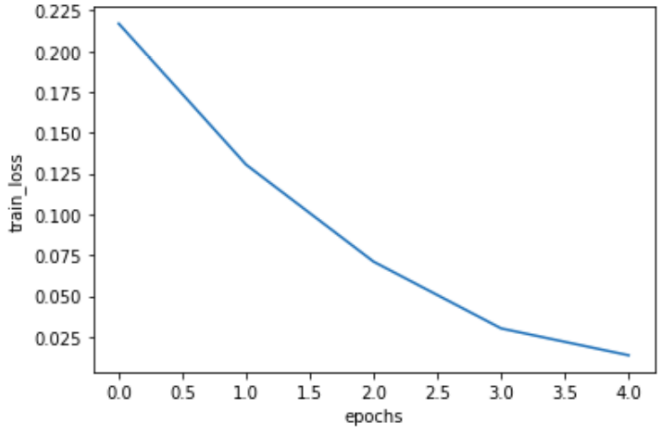
```
de  + Text
    print(f'\t Val. Loss: {valid_loss:.3f} |  Val. Acc: {valid_acc*100:.2f}%')

Epoch: 01 | Epoch Time: 1m 48s
        Train Loss: 0.217 | Train Acc: 86.53%
         Val. Loss: 0.170 |  Val. Acc: 82.25%
Epoch: 02 | Epoch Time: 1m 47s
        Train Loss: 0.130 | Train Acc: 89.65%
         Val. Loss: 0.164 |  Val. Acc: 88.62%
Epoch: 03 | Epoch Time: 1m 47s
        Train Loss: 0.112 | Train Acc: 92.43%
         Val. Loss: 0.150 |  Val. Acc: 90.32%
Epoch: 04 | Epoch Time: 1m 47s
        Train Loss: 0.096 | Train Acc: 94.64%
         Val. Loss: 0.135 |  Val. Acc: 92.29%
Epoch: 05 | Epoch Time: 1m 46s
        Train Loss: 0.079 | Train Acc: 96.60%
         Val. Loss: 0.123 |  Val. Acc: 93.93%
```

Testing Metrics

```
[53] model.load_state_dict(torch.load('tut5-model2.pt'))

     test_loss, test_acc = evaluate(model, test_iterator, criterion)

     print(f'Test Loss: {test_loss:.3f} | Test Acc: {test_acc*100:.2f}%')

     Test Loss: 0.119 | Test Acc: 93.18%
```

Loss Graph

```
[52] import matplotlib.pyplot as plt
     plt.plot(loss_list)
     plt.ylabel('train_loss')
     plt.xlabel('epochs')

     Text(0.5, 0, 'epochs')
```



It was not always at 93 %

- We changed the training data size from **80k to 4 lakh**
- Introduced **learning rate decay**
- Switched from 100 dimension embedding to 200 dimension of **GloVe** so that we have more trainable paramteres.
- Introduced **dropout regularization**
- Increased embedding vocabulary size from **25k to 1 lakh.**

And of course hours of hard work by the team :)

Tech Stack

ML Model

- Pytorch
- GloVe
- Spacy
- Torchtext
- Pandas

BackEnd

- Django
- Celery
- Redis
- PostgreSQL

FrontEnd

- Bootstrap
- HTML
- CSS
- JavaScript

Summary

- **93 %** accuracy with ML model.
- Bulk reviews handled via **job queueing** using **Celery + Redis**
- Tested 77k reviews under 600 seconds of **CPU** time using ngrok.
- Compatible with both GPU and CPU
- Can be easily integrated with the BHUVAN database.
- Can be used via Mobile devices as well. (responsive web design)
- API support can also be provided easily using **Django REST**