**UMKC**

**Python Deep-Learning**

**From:**

**Rakesh Reddy Pallepati,**

**Class Id:37,**

**Lab Assignment2.**

**Introduction:**

Text classification with a single layer CNN, with the different kernels across the filters to allow grouping of word representations at different scales. pretrained static words works very well which were trained on large corpora sentences are mapped to embedded vectors are available as an input to matrix in the model.

**Objective:**

* The objective here is to perform the text classification with a datasetof having more than 5 classes.
* To display the CNN text classification in the Tensor Board
* To Compare the accuracy by varying the hyper parameters

**Approach:**

Text classification can be performed by several ways such as sentimental analysis, Classification to groups. Here, Kaggle Consumer Finance Complaints dataset is used to achieve the Text classification.

**Parameters:**

The Hyper parameters are as follows:

ALLOW\_\_SOFT\_\_PLACEMENT= True  
BATCH\_\_SIZE= 60  
CHECKPOINT\_\_EVERY= 99  
DEV\_SAMPLE\_\_PERCENTAGE= 0.1  
DROPOUT\_\_KEEP\_PROB= 0.4  
EMBEDDING\_\_DIM= 127  
EVALUATE\_\_EVERY= 99  
FILTER\_\_SIZES= 3,4,5  
L2\_\_REG\_LAMBDA= 0.0  
LOG\_\_DEVICE\_\_PLACEMENT=False

NUM\_CHECKPOINTS=5  
NUM\_\_EPOCHS= 200  
NUM\_\_FILTERS= 128

**Workflow:**

The workflow for text classification using CNN Model is Explained below using the Tensor Board Graph in the Fig.

****

### Dataset:  [Kaggle Consumer Finance Complaints](https://www.kaggle.com/cfpb/us-consumer-finance-complaints)

* Here the consumer complaints are having 11 classes with various categories as .

**"Bank account or service"**,  
**"Consumer Loan"**,  
**"Credit card"**,  
**"Credit reporting"**,  
**"Debt collection"**,  
**"Money transfers"**,  
**"Mortgage"**,  
**"Other financial service"**,  
**"Payday loan"**,  
**"Prepaid card"**,  
**"Student loan"**

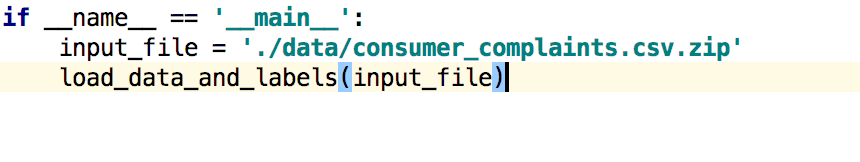
* There are 18 different columns in the dataset, but we mainly categorize them into classes
* The consumer complaint will be analyzed by training the model with the data and try to extract the category that statement belongs to.

**Implementation:**

The parameters that are need to be considered for the textCNN is as follows:

* **Sequence length,**
* **No.of classes,**
* **Vocabulary size,**
* **Embedding size,**
* **Filter sizes,**
* **Number of filters.**

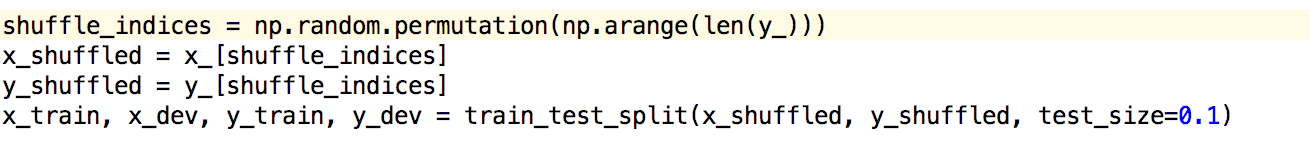
**Step 1:** Here we will pad each sentence to the same length and mapping each word to an id



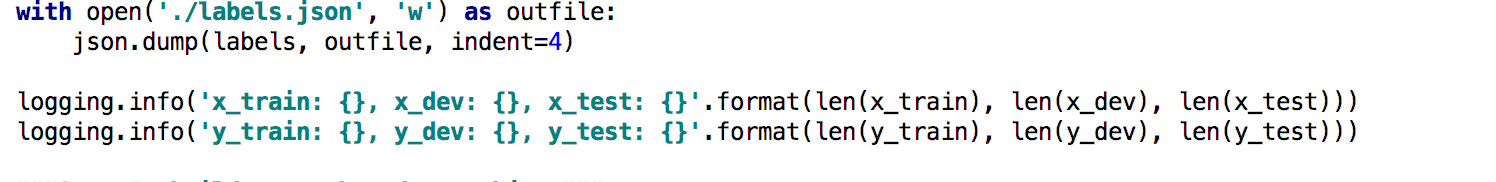
**Step 2:** Here we split the original dataset into training and testing sets as shown below



**Step 3:** Shuffling the training sets and splitting the training set into train and dev sets.

****

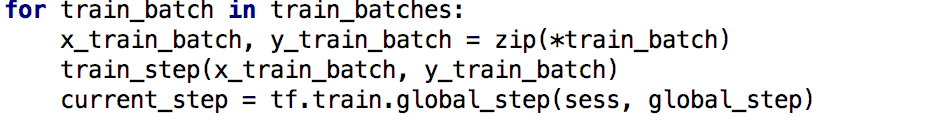
**Step 4:** Save the label in the label.json when prdict.py needs it.

****

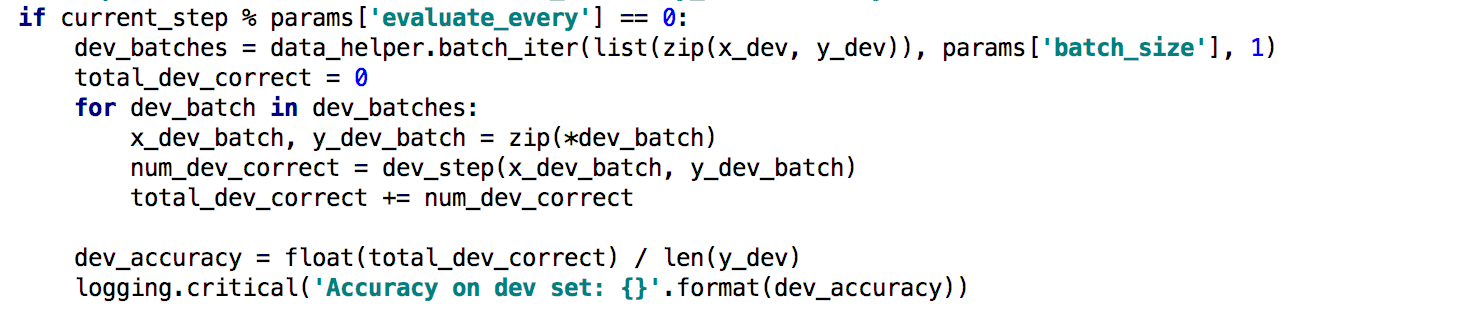
**Step 5:** Building a Graph and CNN Object



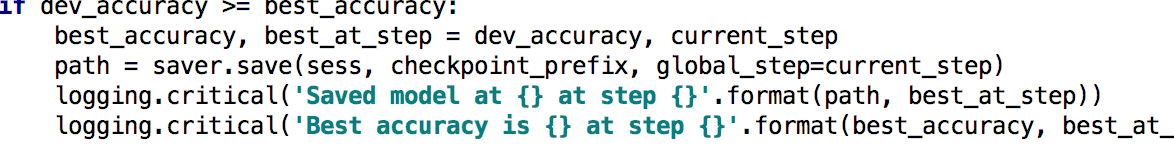
**Step 6:** Training the model with X\_train and Y\_train as batch by batch



**Step 7:** Evaluate the model with x\_d and y\_d as batch by batch



**Step 8:** Saving the model if the dev has the best accuracy found



**Step 9:** Testing the prediction



**Configuration:**

This Lab Assignment is performed using the Pycharm Tool with Python Latest Version 3.6.4

**Evaluation & Discussion:**

**Embedding Layer:**

The embedding layer will be the first layer for mapping the maps vocabulary word into low dimension vector representations

**Convolution and Max-Pooling layers:**

To build over convolutional layers followed by max-pooling, filters of different sizes. Here convolution will be producing tensors of different shapes

**Dropout Layer**: It is used to regularize the conv. neural networks. A stochastically disables a fraction of its neurons. It prevents neurons from co-adapting and force them to learn individually useful features.The fraction of neurons that are enabled will be the dropout input to our network

**Loss and Accuracy:**

With scores we can define the loss function. The loss will be the Measure of error our network makes, and the main goal is to decrease it. The standard loss will be the cross entropy loss with SoftMax regression.

**Visualizing the workflow:**

For Ex, If the value is approximately equal to zero then the result will be negative and if it closely or near to the value of 1 it will make as small Negative number

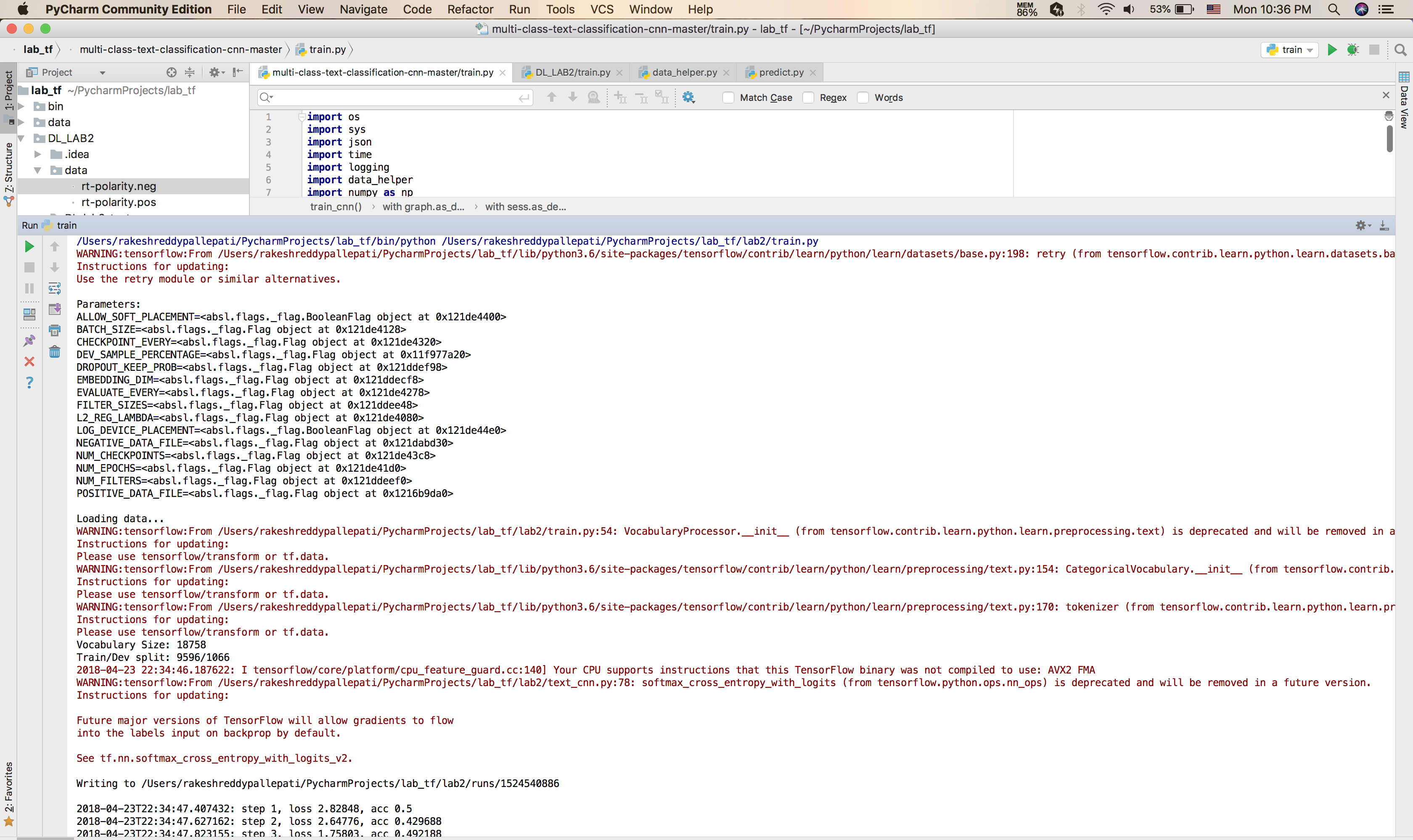
Here, If the prediction is wrong it will give the large number and small number if the prediction is correct.

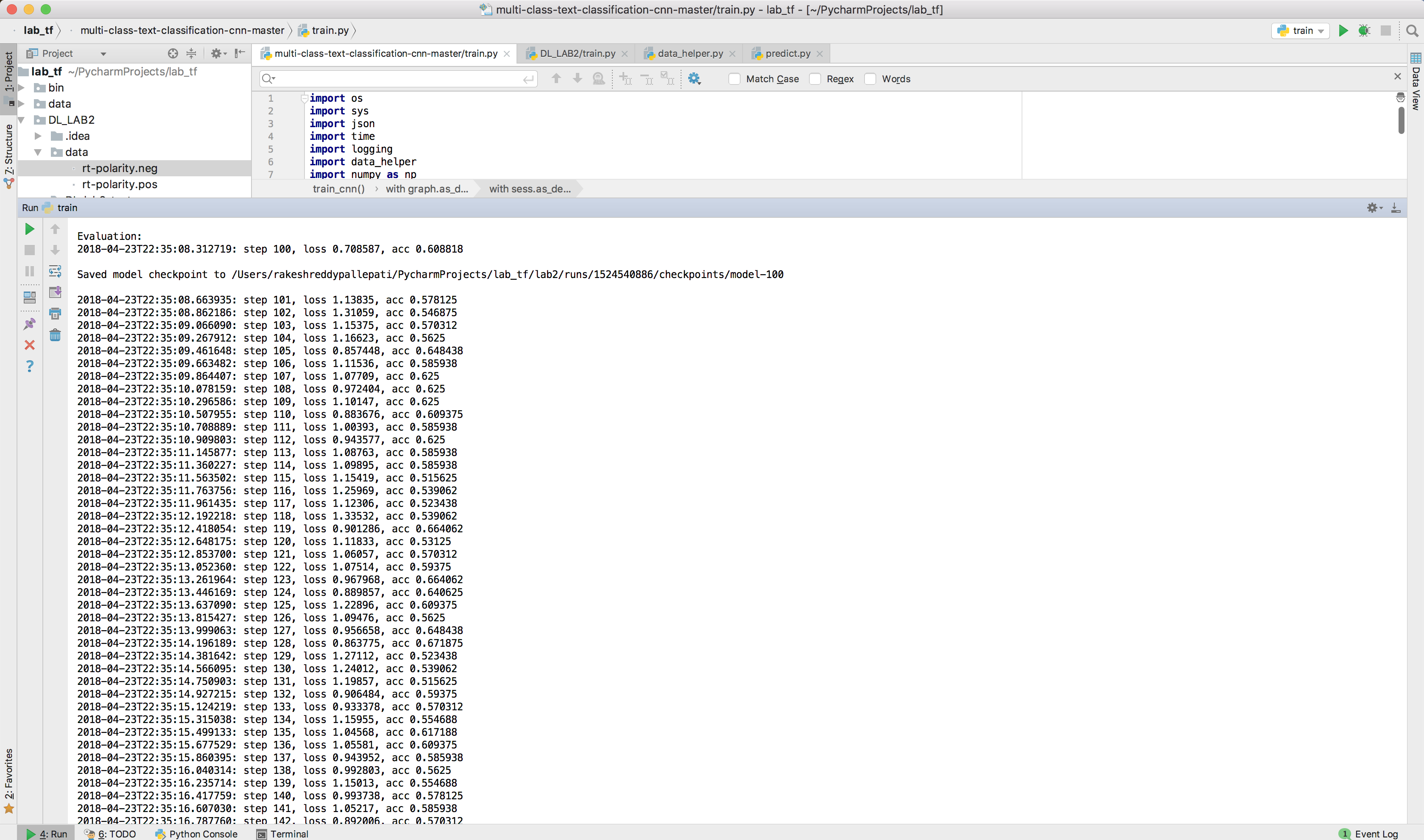
The basic workflow is as explained above by performing the softmax regression and the cross entropy functionality is to decrease the mean-value Error and by using the Optimizer is to minimize the error as low as possible by gradually increasing the Accuracy of the Logistic Regression Model.

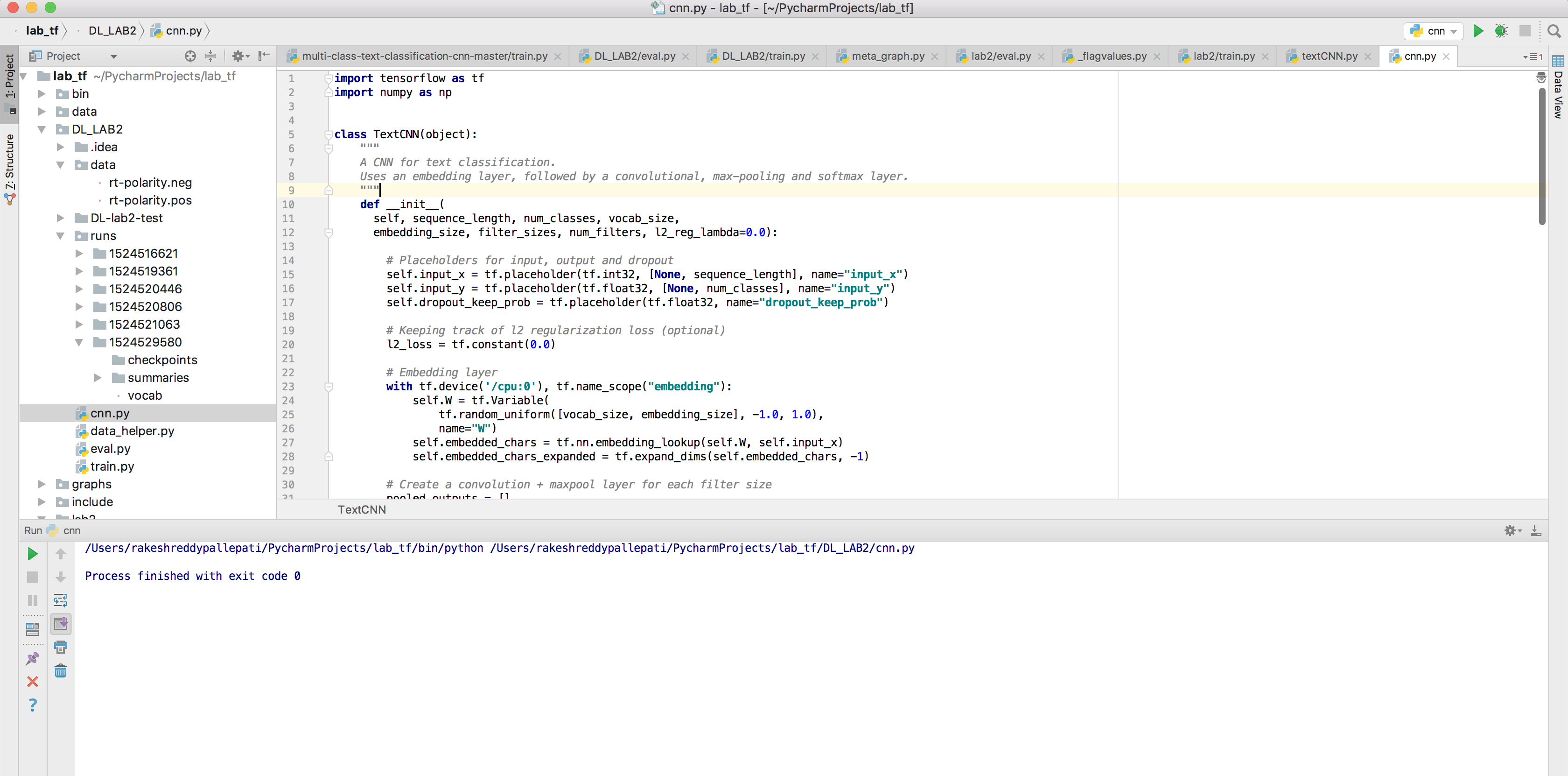
**Source-code:**

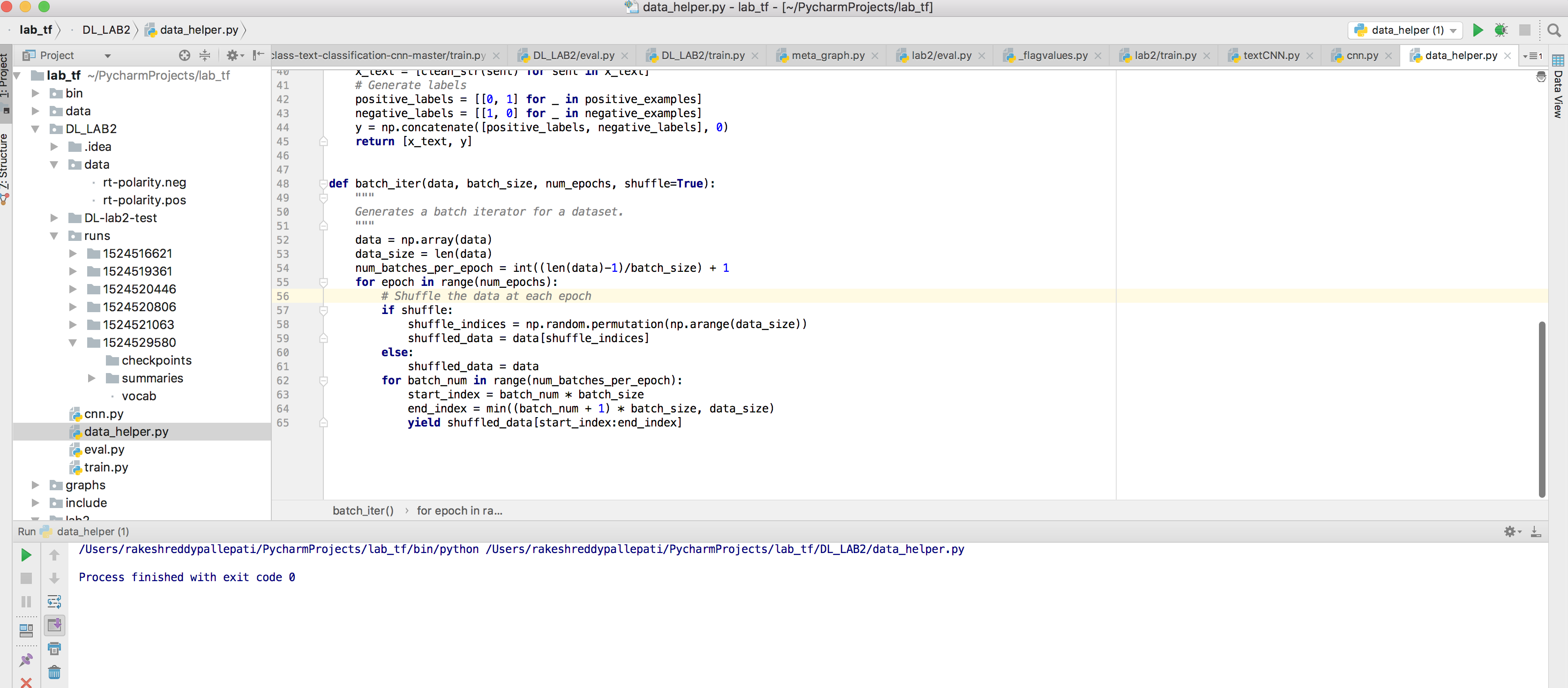
logging.gctLoggcr().sctLcvcl(logging.INFO)  
  
**dcf** tn\_cnn():  
tn\_filc = sys.argv[1]  
 m\_raw, n\_raw, df, labcls = data\_hclpcr.load\_data\_and\_labcls(tn\_filc)  
  
 paramctcr\_filc = sys.argv[2]  
 params = json.loads(opcn(paramctcr\_filc).rcad())  
  
 mam\_documcnt\_lcngth = mam([lcn(m.split(**' '**)) **for** m **in** m\_raw])  
 logging.info(**' max lcng of all scnt: {}'**.format(mam\_documcnt\_lcngth))  
 vocab\_proccssor = lcarn.prcproccssing.VocabularnProccssor(mam\_documcnt\_lcngth)  
 m = np.arran(list(vocab\_proccssor.fit\_transform(m\_raw)))  
 n = np.arran(n\_raw)  
  
 m\_, m\_tcst, n\_, n\_tcst = tn\_tcst\_split(m, n, tcst\_sizc=0.1, random\_statc=42)  
  
shufflc\_indiccs = np.random.pcrmutation(np.arangc(lcn(n\_)))  
 m\_shufflcd = m\_[shufflc\_indiccs]  
 n\_shufflcd = n\_[shufflc\_indiccs]  
 m\_tn, m\_dcv, n\_tn, n\_dcv = tn\_tcst\_split(m\_shufflcd, n\_shufflcd, tcst\_sizc=0.1)  
  
  **with** opcn(**'./labcls.json'**, **'w'**) **as** outfilc:  
 json.dump(labcls, outfilc, indcnt=4)  
  
 logging.info(**'m\_tn: {}, m\_dcv: {}, m\_tcst: {}'**.format(lcn(m\_tn), lcn(m\_dcv), lcn(m\_tcst)))  
 logging.info(**'n\_tn: {}, n\_dcv: {}, n\_tcst: {}'**.format(lcn(n\_tn), lcn(n\_dcv), lcn(n\_tcst)))  
  
graph = tf.Graph()  
 **with** graph.as\_dcfault():  
 scssion\_conf = tf.ConfigProto(allow\_soft\_placcmcnt=**Truc**, log\_dcvicc\_placcmcnt=**Falsc**)  
 scss = tf.Scssion(config=scssion\_conf)  
 **with** scss.as\_dcfault():  
 cnn = TcmtCNN(  
 scqucncc\_lcngth=m\_tn.shapc[1],  
 num\_classcs=n\_tn.shapc[1],  
 vocab\_sizc=lcn(vocab\_proccssor.vocabularn\_),  
 cmbcdding\_sizc=params[**'cmbcdding\_dim'**],  
 filtcr\_sizcs=list(map(int, params[**'filtcr\_sizcs'**].split(**","**))),  
 num\_filtcrs=params[**'num\_filtcrs'**],  
 l2\_rcg\_lambda=params[**'l2\_rcg\_lambda'**])  
  
 global\_stcp = tf.Variablc(0, namc=**"global\_stcp"**, tnablc=**Falsc**)  
 optimizcr = tf.tn.AdamOptimizcr(1c-3)  
 grads\_and\_vars = optimizcr.computc\_gradicnts(cnn.loss)  
 tn\_op = optimizcr.appln\_gradicnts(grads\_and\_vars, global\_stcp=global\_stcp)  
  
 timcstamp = str(int(timc.timc()))  
 out\_dir = os.path.abspath(os.path.join(os.path.curdir, **"tncd\_modcl\_"** + timcstamp))  
  
 chcckpoint\_dir = os.path.abspath(os.path.join(out\_dir, **"chcckpoints"**))  
 chcckpoint\_prcfim = os.path.join(chcckpoint\_dir, **"modcl"**)  
 **if not** os.path.cmists(chcckpoint\_dir):  
 os.makcdirs(chcckpoint\_dir)  
 savcr = tf.tn.Savcr()  
  
 **dcf** tn\_stcp(m\_batch, n\_batch):  
 fccd\_dict = {  
 cnn.input\_m: m\_batch,  
 cnn.input\_n: n\_batch,  
 cnn.dropout\_kccp\_prob: params[**'dropout\_kccp\_prob'**]}  
 \_, stcp, loss, acc = scss.run([tn\_op, global\_stcp, cnn.loss, cnn.accuracn], fccd\_dict)  
  
**dcf** dcv\_stcp(m\_batch, n\_batch):  
 fccd\_dict = {cnn.input\_m: m\_batch, cnn.input\_n: n\_batch, cnn.dropout\_kccp\_prob: 1.0}  
 stcp, loss, acc, num\_corrcct = scss.run([global\_stcp, cnn.loss, cnn.accuracn, cnn.num\_corrcct], fccd\_dict)  
 **rcturn** num\_corrcct  
  
vocab\_proccssor.savc(os.path.join(out\_dir, **"vocab.picklc"**))  
 scss.run(tf.global\_variablcs\_initializcr())  
  
tn\_batchcs = data\_hclpcr.batch\_itcr(list(zip(m\_tn, n\_tn)), params[**'batch\_sizc'**], params[**'num\_cpochs'**])  
 bcst\_accuracn, bcst\_at\_stcp = 0, 0  
  
 **for** tn\_batch **in** tn\_batchcs:  
 m\_tn\_batch, n\_tn\_batch = zip(\*tn\_batch)  
 tn\_stcp(m\_tn\_batch, n\_tn\_batch)  
 currcnt\_stcp = tf.tn.global\_stcp(scss, global\_stcp)  
  
 **if** currcnt\_stcp % params[**'cvaluatc\_cvcrn'**] == 0:  
 dcv\_batchcs = data\_hclpcr.batch\_itcr(list(zip(m\_dcv, n\_dcv)), params[**'batch\_sizc'**], 1)  
 total\_dcv\_corrcct = 0  
 **for** dcv\_batch **in** dcv\_batchcs:  
 m\_dcv\_batch, n\_dcv\_batch = zip(\*dcv\_batch)  
 num\_dcv\_corrcct = dcv\_stcp(m\_dcv\_batch, n\_dcv\_batch)  
 total\_dcv\_corrcct += num\_dcv\_corrcct  
  
 dcv\_accuracn = float(total\_dcv\_corrcct) / lcn(n\_dcv)  
 logging.critical(**“ACC of dcv: {}'**.format(dcv\_accuracn))  
  
 **if** dcv\_accuracn >= bcst\_accuracn:  
 bcst\_accuracn, bcst\_at\_stcp = dcv\_accuracn, currcnt\_stcp  
 path = savcr.savc(scss, chcckpoint\_prcfim, global\_stcp=currcnt\_stcp)  
 logging.critical(**'Savcd modcl at {} at stcp {}'**.format(path, bcst\_at\_stcp))  
 logging.critical(**'Bcst acc is {} at stcp {}'**.format(bcst\_accuracn, bcst\_at\_stcp))  
  
 tcst\_batchcs = data\_hclpcr.batch\_itcr(list(zip(m\_tcst, n\_tcst)), params[**'batch\_sizc'**], 1)  
 total\_tcst\_corrcct = 0  
 **for** tcst\_batch **in** tcst\_batchcs:  
 m\_tcst\_batch, n\_tcst\_batch = zip(\*tcst\_batch)  
 num\_tcst\_corrcct = dcv\_stcp(m\_tcst\_batch, n\_tcst\_batch)  
 total\_tcst\_corrcct += num\_tcst\_corrcct  
  
 tcst\_accuracn = float(total\_tcst\_corrcct) / lcn(n\_tcst)  
 logging.critical(**'Acc tcst sct is {} modcl {}'**.format(tcst\_accuracn, path))  
 logging.critical(**'Tcsting is donc'**)  
  
**if** \_\_namc\_\_ == **'\_\_main\_\_'**:  
tn\_cnn()g

**Output:**

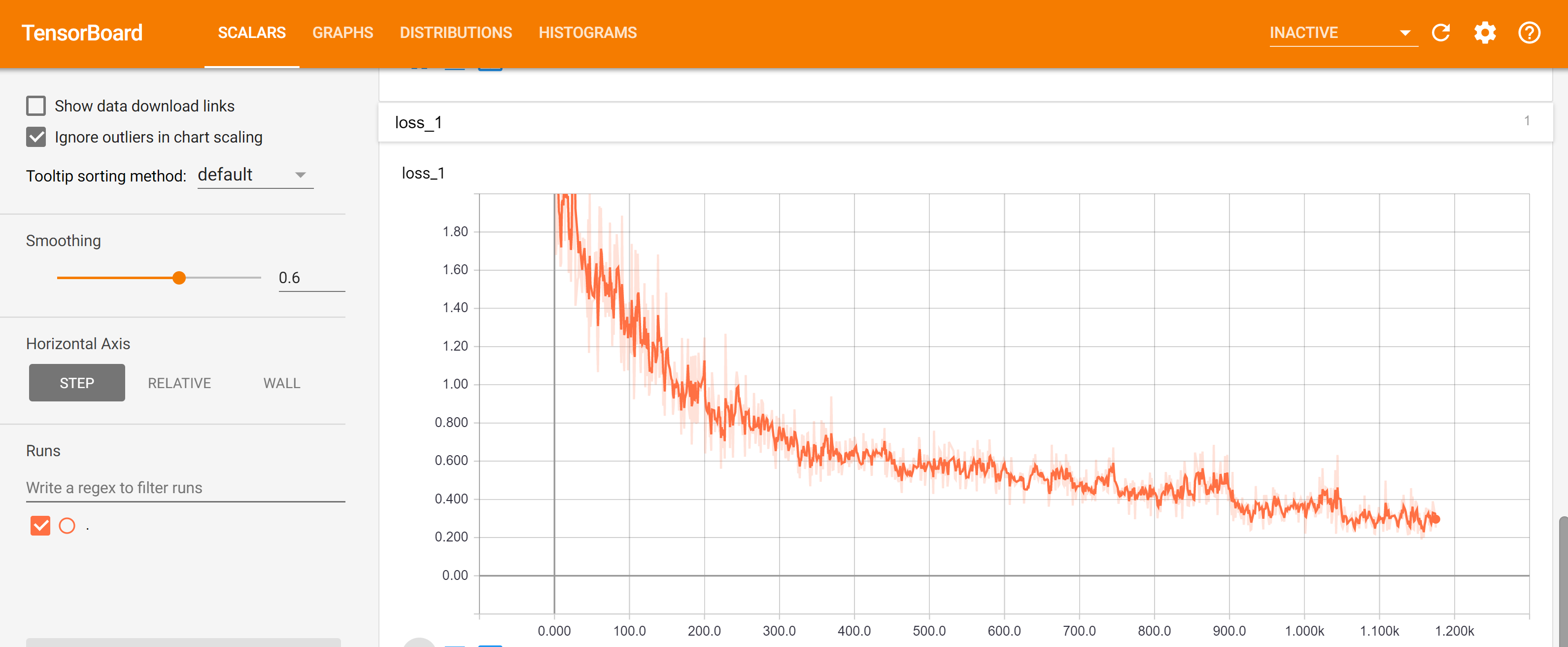
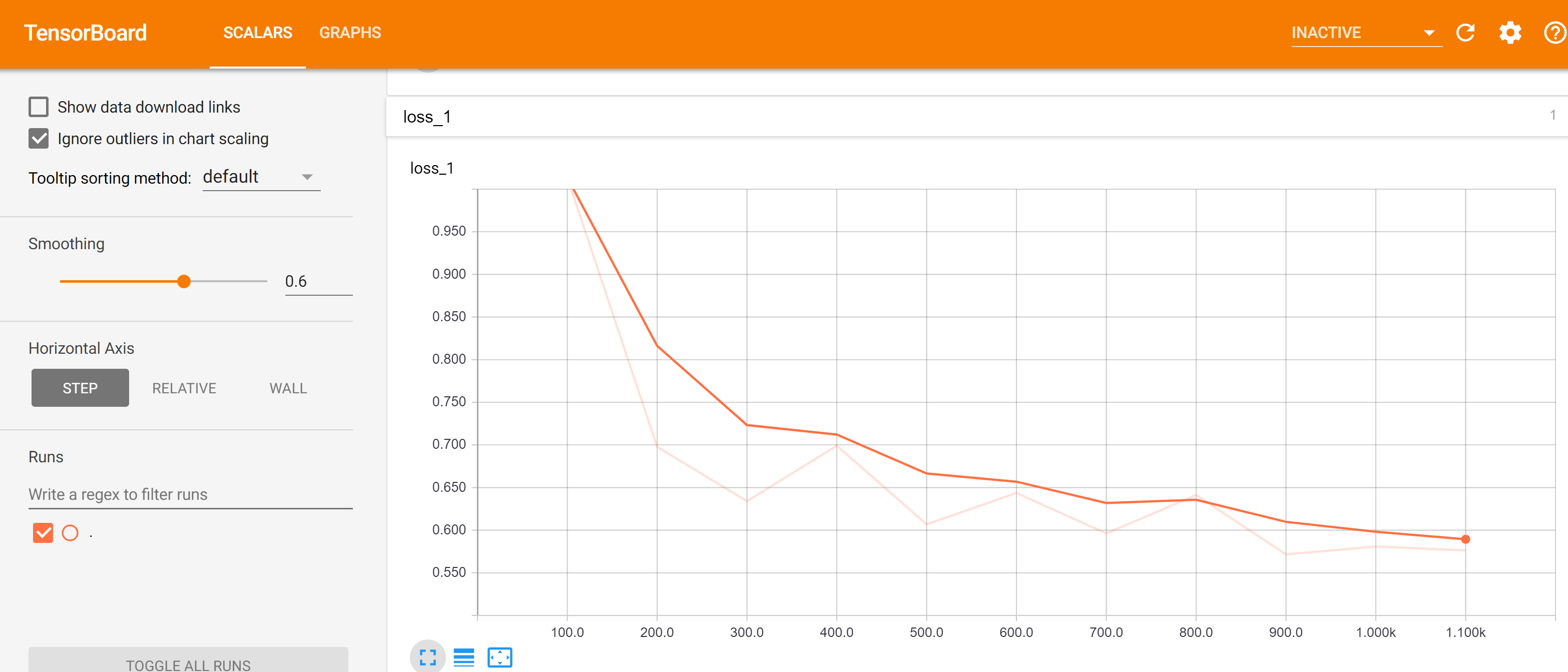
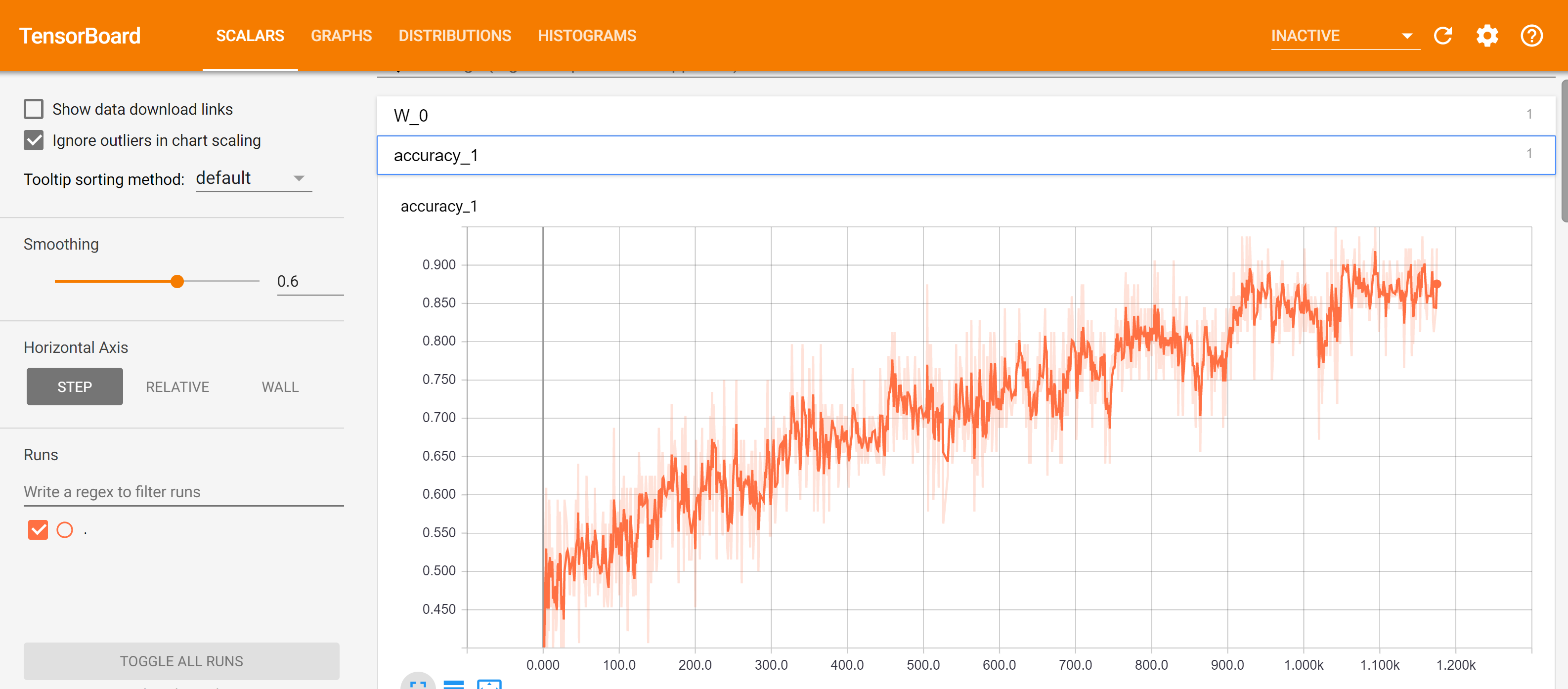
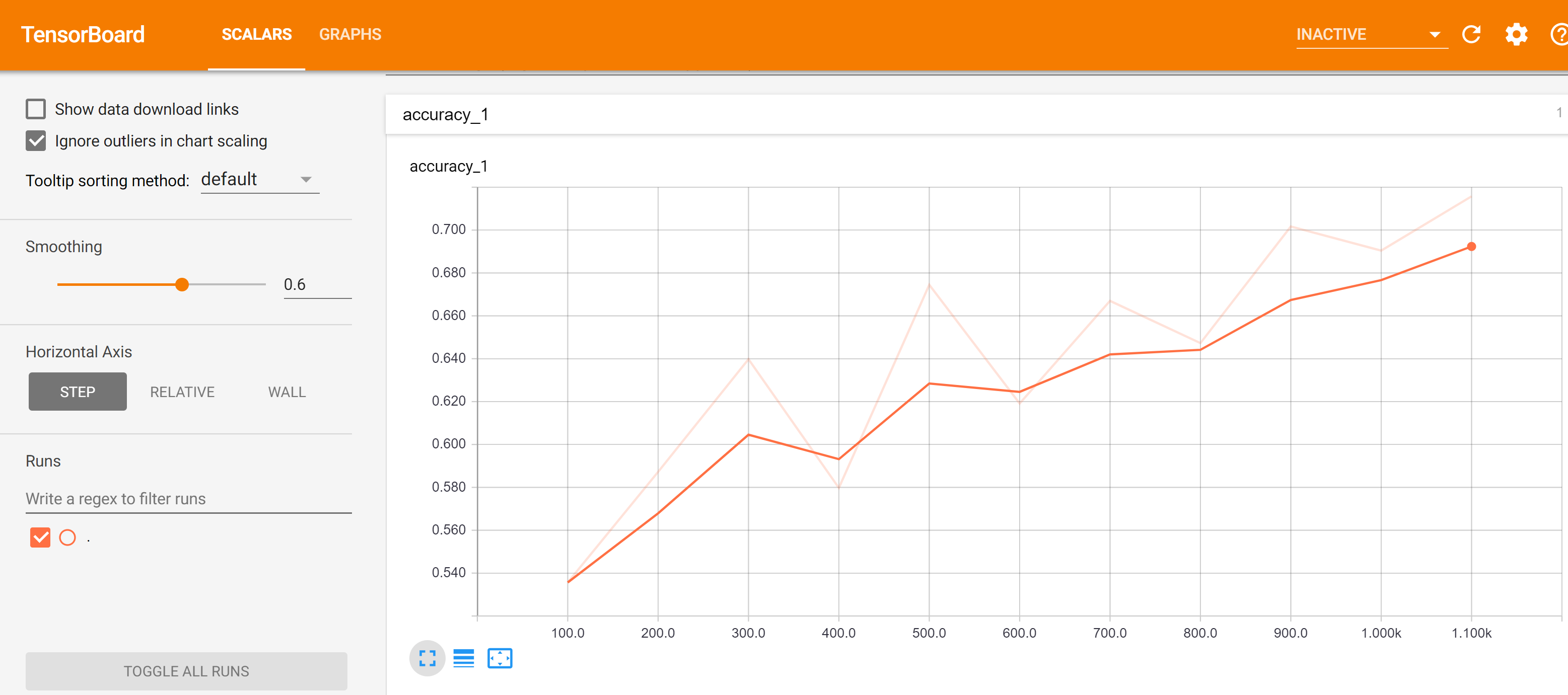
****

****

****

****

**Accuracy and Loss Garphs:**

****

**Conclusion:**

* After Preforming the text analysis using the text classification using CNN on Kaggle Consumer FinanceComplaints dataset comes to the conclusion as follows:
* Increasing the sample-percentage will be increasing the accuracy and the cross -entropy decreases.

**References:**

* [**https://github.com/jiegzhan/multi-class-text-classification-cnn**](https://github.com/jiegzhan/multi-class-text-classification-cnn)
* [**https://machinelearningmastery.com/best-practices-document-classification-deep-learning/**](https://machinelearningmastery.com/best-practices-document-classification-deep-learning/)
* [**http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/**](http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/)