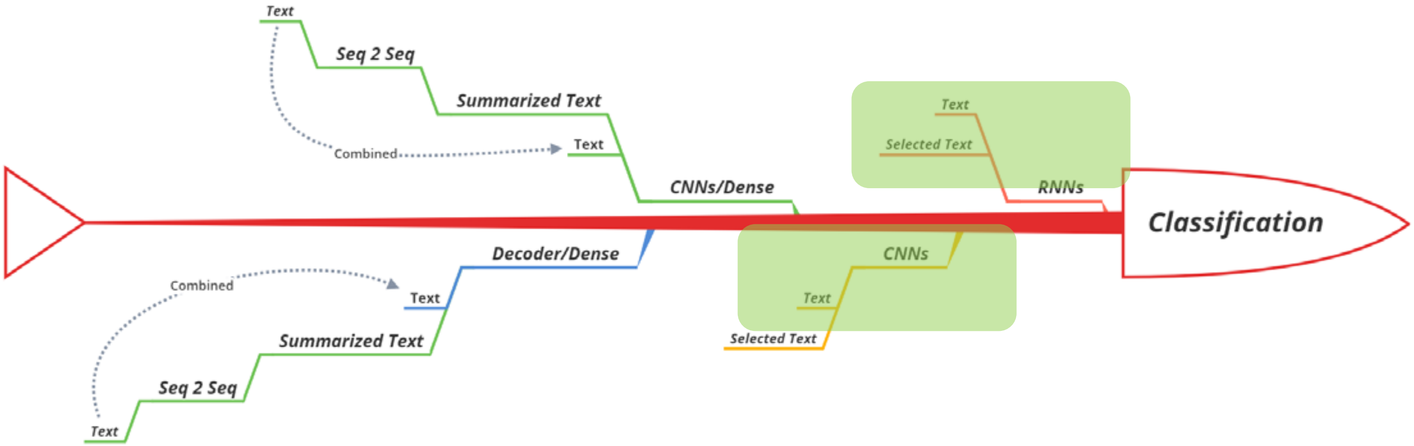
Tweet Sentiment Extraction

Linsen Li
Wei Yang

Outline

- Problem description
- Text classification part
  - RNNs
  - CNNs



- Text summarization part
  - Attention Seq to Seq
- Mid-stage conclusion
- Further study

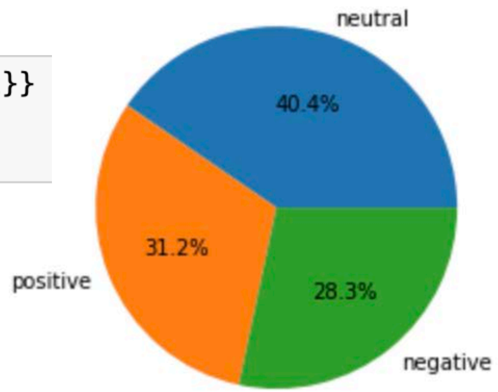
Problem description

The data set and label proportion

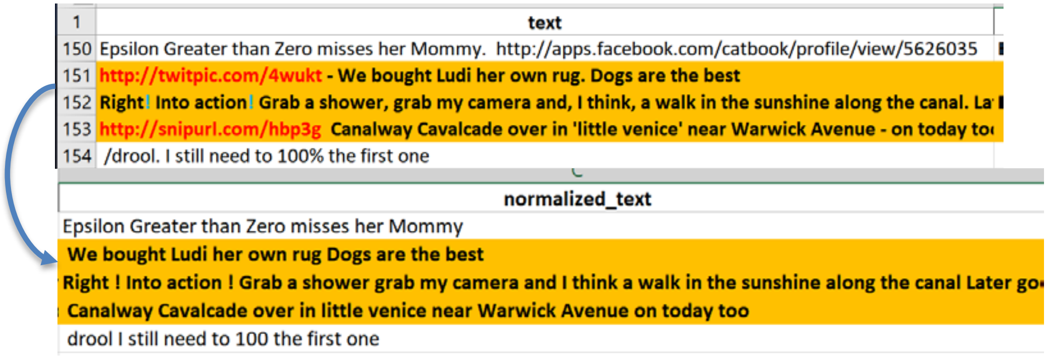
	textID	text	selected_text	sentiment
0	a3d0a7d5ad	Spent the entire morning in a meeting w/ a ven...	my boss was not happy w/ them. Lots of fun.	0
1	251b6a6766	Oh! Good idea about putting them on ice cream	Good	1
2	c9e8d1ef1c	says good (or should i say bad?) afternoon! h...	says good (or should i say bad?) afternoon!	0
3	f14f087215	i dont think you can vote anymore! i tried	i dont think you can vote anymore!	2
4	bf7473b12d	haha better drunken tweeting you mean?	better	1
...	...	...	...	...
27481	3dbae74fcd	I want to go to VP, but no one is willing to c...	I want to go to VP, but no one is willing to c...	0
27482	63147b35cb	Wah, why are you sad?	Wah, why are you sad?	0
27483	bdb196a09f	playing sudoku while mommy makes me breakfast ...	playing sudoku while mommy makes me breakfast ...	0
27484	18c2a1e98e	see u bye see u! i love the hot30	i love	1
27485	1c1f3724db	ha ha, and what game is that? i like games	? i like	

27485 rows × 4 columns

```
replace_map = {'sentiment': {'neutral': 0, 'positive': 1, 'negative': 2}}
train.replace(replace_map, inplace=True)
test.replace(replace_map, inplace=True)
```



Preprocessing & EDA

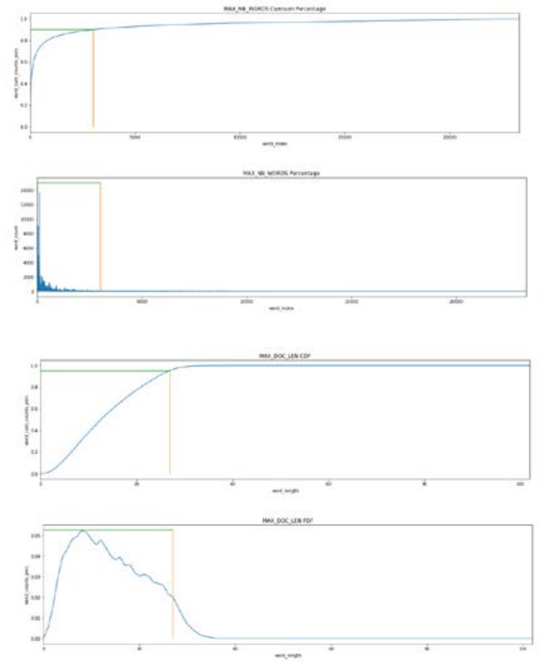


1	text
150	Epsilon Greater than Zero misses her Mommy. http://apps.facebook.com/catbook/profile/view/5626035
151	http://twitpic.com/4wukt - We bought Ludi her own rug. Dogs are the best
152	Right! Into action! Grab a shower, grab my camera and, I think, a walk in the sunshine along the canal. La
153	http://snipurl.com/hbp3g Canalway Cavalcade over in 'little venice' near Warwick Avenue - on today to
154	/drool. I still need to 100% the first one

normalized_text
Epsilon Greater than Zero misses her Mommy
We bought Ludi her own rug Dogs are the best
Right ! Into action ! Grab a shower grab my camera and I think a walk in the sunshine along the canal Later go
Canalway Cavalcade over in little venice near Warwick Avenue on today too
drool I still need to 100 the first one

To reduce noise

- Website deleted
- Treat “?!” as single word (Important)



To improve efficiency

- 90% of MAX_NB_WORDS = 3014
    - 95% of MAX_NB_WORDS = 7834
- 95% of DOC_LEN = 27

Text classification - RNNs

RNN for text classification

- Training set, validation set split
- Tokenization and text to sequence
- Set the length of input=100 for text column

```
Shape of x_train: (24736, 100)
Shape of y_train: (24736, 3)
Shape of x_validation: (2749, 100)
Shape of y_vallidation: (2749, 3)
```

- Set the length of input=50 for selected_text column

```
Shape of x_train: (24736, 50)
Shape of y_train: (24736, 3)
Shape of x_validation: (2749, 50)
Shape of y_vallidation: (2749, 3)
```

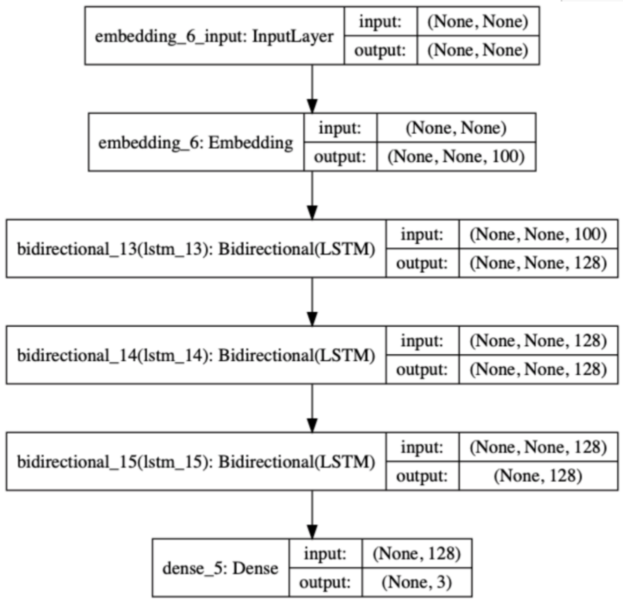
Text classification - RNNs

RNN for text classification

- Use LSTM instead of Simple RNN
- Use stacked LSTM
- Use bidirectional LSTM
- Use pre-train word embedding

```
from keras.models import Sequential
from keras.layers import LSTM, Embedding, Dense, Bidirectional


state_dim =64
embedding_dim=100
model_lstm = Sequential()
model_lstm.add(Embedding(vocab_size, embedding_dim, weights=[embedding_matrix], trainable=False))
model_lstm.add(Bidirectional(LSTM(state_dim,return_sequences=True,dropout=0.5,recurrent_dropout=0.5)))
model_lstm.add(Bidirectional(LSTM(state_dim,return_sequences=True,dropout=0.5,recurrent_dropout=0.5)))
model_lstm.add(Bidirectional(LSTM(state_dim,return_sequences=False,dropout=0.5,recurrent_dropout=0.5)))
model_lstm.add(Dense(3,activation='softmax'))
```

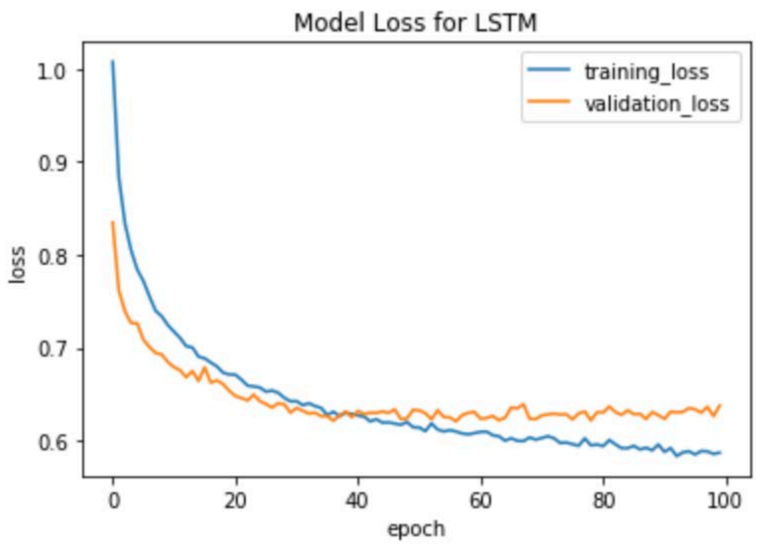
70]:




```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_15 (Embedding)     (None, None, 100)         2533400
_________________________________________________________________
bidirectional_25 (Bidirectio (None, None, 128)         84480
_________________________________________________________________
bidirectional_26 (Bidirectio (None, None, 128)         98816
_________________________________________________________________
bidirectional_27 (Bidirectio (None, 128)               98816
_________________________________________________________________
dense_9 (Dense)              (None, 3)                 387
=================================================================
Total params: 2,815,899
Trainable params: 282,499
Non-trainable params: 2,533,400
_________________________________________________________________
```

Text classification - RNNs

RNN result for text column




```
loss_and_acc = model_lstm.evaluate(x_test,y_test_c,verbose=0)
print('loss = ' + str(loss_and_acc[0]))
print('accuracy = ' + str(loss_and_acc[1]))
```

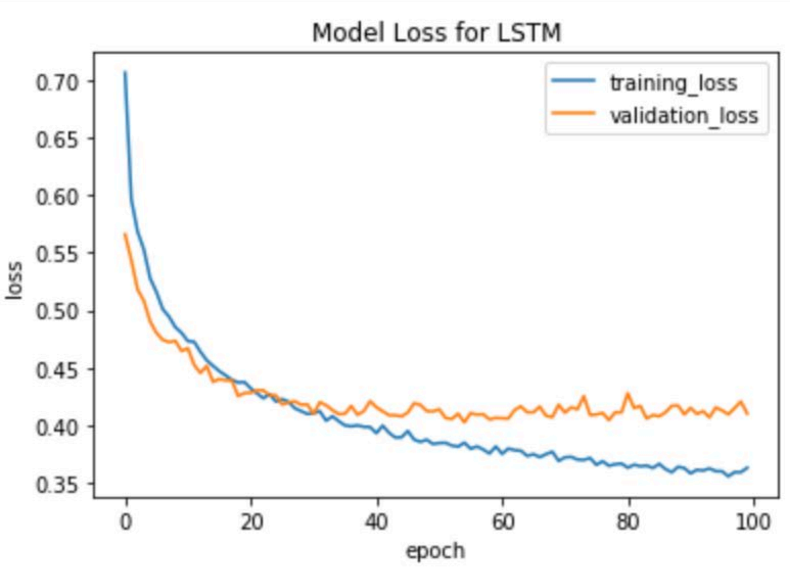
```
loss = 0.5693516473459651
accuracy = 0.7686431407928467
```

```
# Report the recall and precision for each category on the test set
from sklearn.metrics import classification_report
y_pred_NN = model_lstm.predict([x_test], batch_size=16, verbose=0)
y_pred_bool = np.argmax(y_pred_NN, axis=1)
print(classification_report(y_test, y_pred_bool))
```

	precision	recall	f1-score	support
0	0.78	0.82	0.80	1133
1	0.82	0.74	0.78	882
2	0.70	0.72	0.71	734
accuracy			0.77	2749
macro avg	0.77	0.76	0.76	2749
weighted avg	0.77	0.77	0.77	2749

Text classification - RNNs

RNN result for selected_text column




```
loss_and_acc = model_lstm1.evaluate(x_test,y_test_c)
print('loss = ' + str(loss_and_acc[0]))
print('accuracy = ' + str(loss_and_acc[1]))
```

```
2749/2749 [==============================] - 4s 1ms/step
loss = 0.4104320356285585
accuracy = 0.8410331010818481
```

```
# Report the recall and precision for each category on the test s
from sklearn.metrics import classification_report
y_pred_NN = model_lstm1.predict([x_test], batch_size=16, verbose=
y_pred_bool = np.argmax(y_pred_NN, axis=1)
print(classification_report(y_test, y_pred_bool))
```

```
2749/2749 [==============================] - 4s 1ms/step
              precision    recall  f1-score   support

           0       0.85      0.85      0.85      1133
           1       0.86      0.84      0.85       882
           2       0.80      0.82      0.81       734

    accuracy                           0.84      2749
   macro avg       0.84      0.84      0.84      2749
weighted avg       0.84      0.84      0.84      2749
```

Text classification - RNNs

RNN result conclusion

- For both text column and selected_text column, the negative label prediction has the worst behavior
- The selected_text absolutely behaves better than text in sentiment classification

Text classification - CNNs

- Local features extraction
  - Multi-channels text CNNs
  - Adjustable channels (Like N-gram + key words extraction)
- Classification task
  - Dense - (Filter size) x (Number of filter)
  - Dropout - (0.4)
  - Half Dense
  - Softmax

Text classification - CNNs

Grid Search conclusion

mean_fit_time	param_EMB	param_FS	param_NF	param_optimizer	mean_test_score	std_test_score	rank_test_	mean_train_score	std_train_score
159.6531413	100	(2, 3, 4, 5)	32	**<keras.optimizers.Adam**	**0.826860106**	**0.001163336**	1	0.931507714	0.011877416
113.4938084	100	(2, 3, 4, 5)	32	Adadelta	**0.824722576**	**0.000750092**	2	0.978442799	0.000589024
218.0452211	100	(2, 3, 4, 5)	32	Adamax	0.822585046	0.001611405	3	0.973349003	0.003029481
105.1464674	100	(2, 3, 4, 5)	32	Adagrad	0.815899582	0.001710644	4	0.978783856	0.000843584
104.1326965	100	(2, 3, 4, 5)	32	RMSprop	0.813489176	0.003711631	5	**0.983104401**	0.000834658
101.6873125	100	(2, 3, 4, 5)	32	<keras.optimizers.RMSprc	0.811169729	0.002640678	6	0.850782242	0.000701435
123.3534462	100	(2, 3, 4, 5)	32	Adam	0.806667273	0.002336764	7	0.986151536	5.61E-05
247.9763242	100	(2, 3, 4, 5)	32	Nadam	0.798799345	0.011473961	8	0.972507975	0.017012146
99.13591552	100	(2, 3, 4, 5)	32	<keras.optimizers.SGD ob	0.790431144	0.027579446	9	0.889984287	0.037769107
97.97679551	100	(2, 3, 4, 5)	32	SGD	0.589821721	0.00430279	10	0.591572634	0.002328828

- Early stopping to avoid overfit
  - Monitor validation accuracy, mode=max
  - Patience = 8
- Grid Search + 3 fold Cross Validation
  - Optimizers = adam (lr=0.0001)
  - Filter size = [2,3,4,5] – Four channels
  - Number of Filters = 24 or 32
  - Embedding dimension = 200
  - Pretrained embedding
    - CBOW
    - Glove

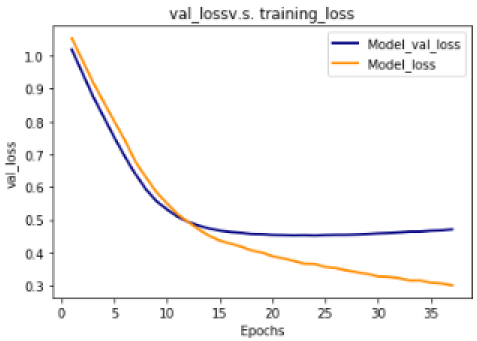
```python
# define the grid search parameters
BestModel_Name = 'GS_model'
EMB =[100]
NF =[32]
FS = [(2,3,4,5)]
MDL= [MAX_DOC_LEN]
MNW= [MAX_NB_WORDS]
PWV = [None]
trainable_switch = [True]
sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
adam = optimizers.adam(lr=1e-4)
rmsprop = optimizers.rmsprop(lr=1e-4)
optimizer = [sgd, 'SGD', rmsprop, 'RMSprop', 'Adagrad', 'Adadelta', adam, 'Adam', 'Adamax', 'Nadam']
earlyStopping = EarlyStopping(monitor='acc', patience=patience, verbose=2, mode='max') # patience: number of epochs with n
callbacks=[earlyStopping]

model = KerasClassifier(build_fn=model_Create, epochs=epoch, batch_size=batch_size, verbose=0)# , callbacks=earlyStopping)
param_grid =dict(EMB=EMB, NF=NF, FS=FS, MDL=MDL, MNW=MNW, PWV=PWV,trainable_switch=trainable_switch, optimizer=optimizer)
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=n_jobs, cv=3, return_train_score=True, verbose=3)
grid_result = grid.fit(x_train, y_train, callbacks=callbacks)
```

Text classification - CNNs

CNN result - **Checkpoint** to load back the best model

```
def train_model(model, x_train, y_train, x_test, y_test, BATCH_SIZE, NUM_EPOCHES, BestModel_Name="best_model" ):
    #### Best model load back
    patience=10
    BEST_MODEL_FILEPATH = BestModel_Name
    earlyStopping = EarlyStopping(monitor='val_loss', patience=patience, verbose=1, mode='min') # patience: number of epochs with no improvement on monitor : val_loss
    checkpoint = ModelCheckpoint(BEST_MODEL_FILEPATH, monitor='val_loss', verbose=0, save_best_only=True, mode='min')
    history = model.fit(x_train, y_train, validation_split=0.2, batch_size=BATCH_SIZE, epochs=NUM_EPOCHES, callbacks=[earlyStopping, checkpoint], verbose=2)
    model.load_weights(BestModel_Name)
```

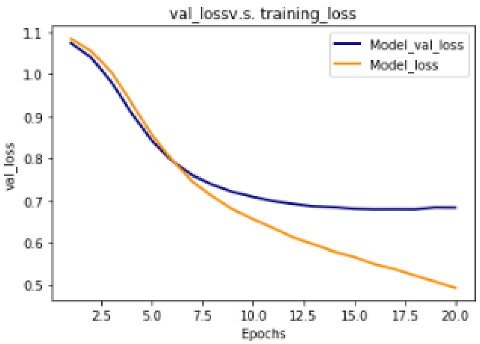
- Selected Text → sentiment



	precision	recall	f1-score	support
0	0.88	0.79	0.83	857
1	0.82	0.83	0.83	1112
2	0.79	0.80	0.79	780
micro avg	0.83	0.81	0.82	2749
macro avg	0.83	0.80	0.82	2749
weighted avg	0.83	0.81	0.82	2749
samples avg	0.81	0.81	0.81	2749

acc: 82.18%

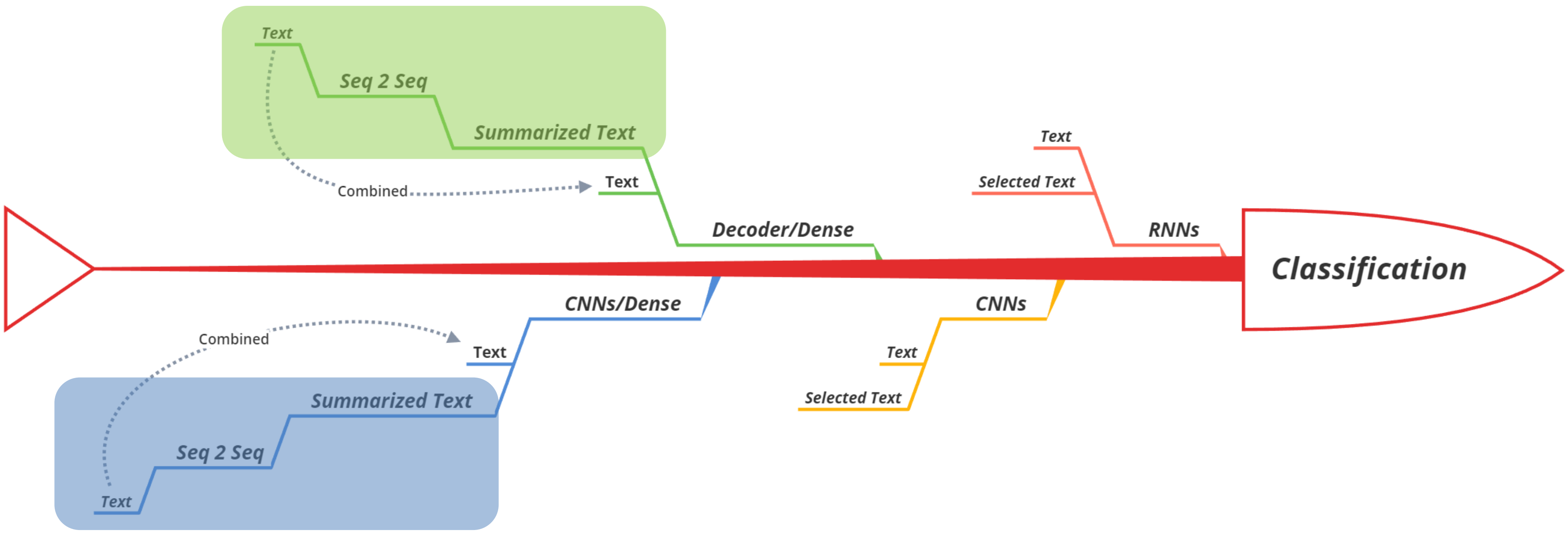
- Text → sentiment



	precision	recall	f1-score	support
0	0.83	0.74	0.78	857
1	0.69	0.68	0.68	1112
2	0.74	0.63	0.68	780
micro avg	0.74	0.69	0.71	2749
macro avg	0.75	0.69	0.72	2749
weighted avg	0.75	0.69	0.71	2749
samples avg	0.69	0.69	0.69	2749

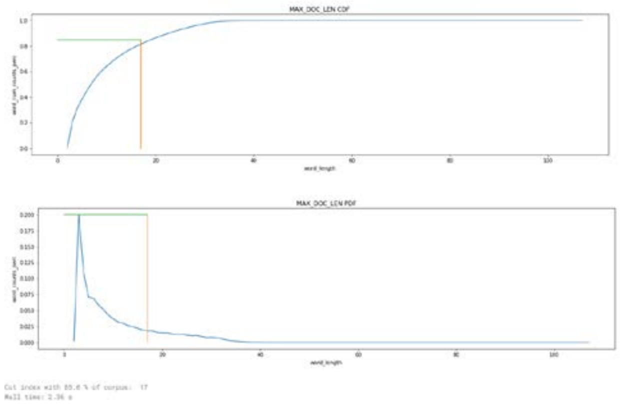
acc: 72.03%

Classification Target



Text summarization

Little Difference Hyper-parameters



- Preprocessing
  - keep the "."
  - Only 85% of DOC_LEN = 17 (Decoder Input)

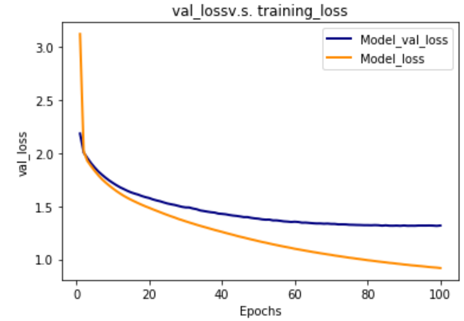
- Training Model Setting
  - Latent dimension = 64
  - Embedding = 200
  - Attention score layer:
    - Activity Regularize = L1 Norm (0.005)
    - Focus on local word as context to do the summarization

```
############ Attention part
encoder_outputs_transforming = Dense(latent_dim, activation=None, use_bias=False, name='encoder_outputs_latent_dim_Squeez')# Transform encoder ou
encoder_outputs_transformed = encoder_outputs_transforming(encoder_outputs)
attention = Dot(axes = [2,2])([decoder_lstm_output, encoder_outputs_transformed]) # Calculate alignment between decoder and encoder
attention = Activation('tanh', name='tanh')(attention) # it depends on which kind of attention we use
alignment = Activation('softmax', name='softmax')(attention)# Normalize alignment score
context = Dot(axes=[2,1]) ([alignment, encoder_outputs]) # Weighted sum of encoder sequence hidden states = [Query, values]
decoder_combined_context = Concatenate(axis = -1)([context, decoder_lstm_output]) # Concatenate context with decoder output as feature
############ Prediction part
attention_outputing = Dense(max_fr_words, activation='softmax', name='attention_score', activity_regularizer = regularizers.l1(0.005)) # he acti
decoder_outputs = attention_outputing(decoder_combined_context) #  This is time distri
```

Text summarization

Summarization result

```
=====================================================================
val_loss loss
```



- S2S BELU score
  - 0.045
  - 0.104 (Additive Attention)
- Result Comparation

```
------------------------------
Sentiment negative
Original Text:  just realized that Chris Lake was spinning in stockholm yesterday, and I missed it!
Ground-truth Selected_text:  missed
Summary from seq2seq model:  just realised that a nap on love
Summary from seq2seq + Attention model:  missed

------------------------------
Sentiment positive
Original Text:  is sipping OJ in the sun in San Pedro at La Soberana again, with sunny smiley Nita...  The garlic tomato paste is delicious! Yummy!!!
Ground-truth Selected_text:  Yummy!!!
Summary from seq2seq model:  s day in the night .
Summary from seq2seq + Attention model:  the delicious ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! !

------------------------------
Sentiment neutral
Original Text:  home sweet home  sleeping until monday I hope.
Ground-truth Selected_text:  home sweet home  sleeping until monday I hope.
Summary from seq2seq model:  home sweet dreams home i m gonna get ready to be watching again
Summary from seq2seq + Attention model:  tired
```

To be done

- Tune the parameters for Attention S2S Model
  - Evaluate based on BLEU score
  - Bidirectional LSTM will be added
- Feature combination - Concatenate / Summation
  - Combined original input ‘text’ embedding
  - The output from Decoder features (contextual embedding)
- Classification
  - Dense

- BERT pretrained Embedding + Dense

STEVENS

INSTITUTE of TECHNOLOGY

THE INNOVATION UNIVERSITY®

stevens.edu