Text Classification

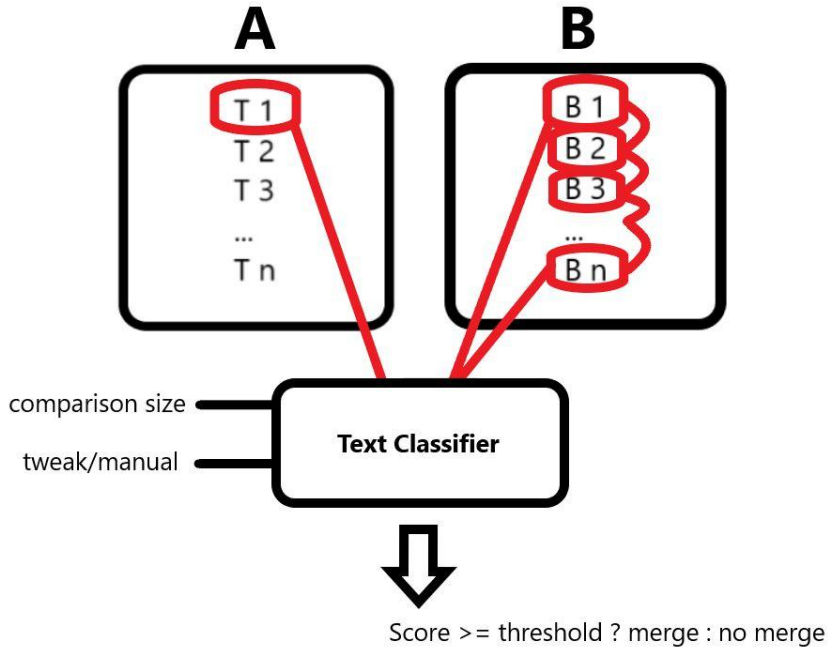
Seeger Zou

```
"title": "Are there any accessible parking available?",
"body": "It has street and private lot parking.",
```

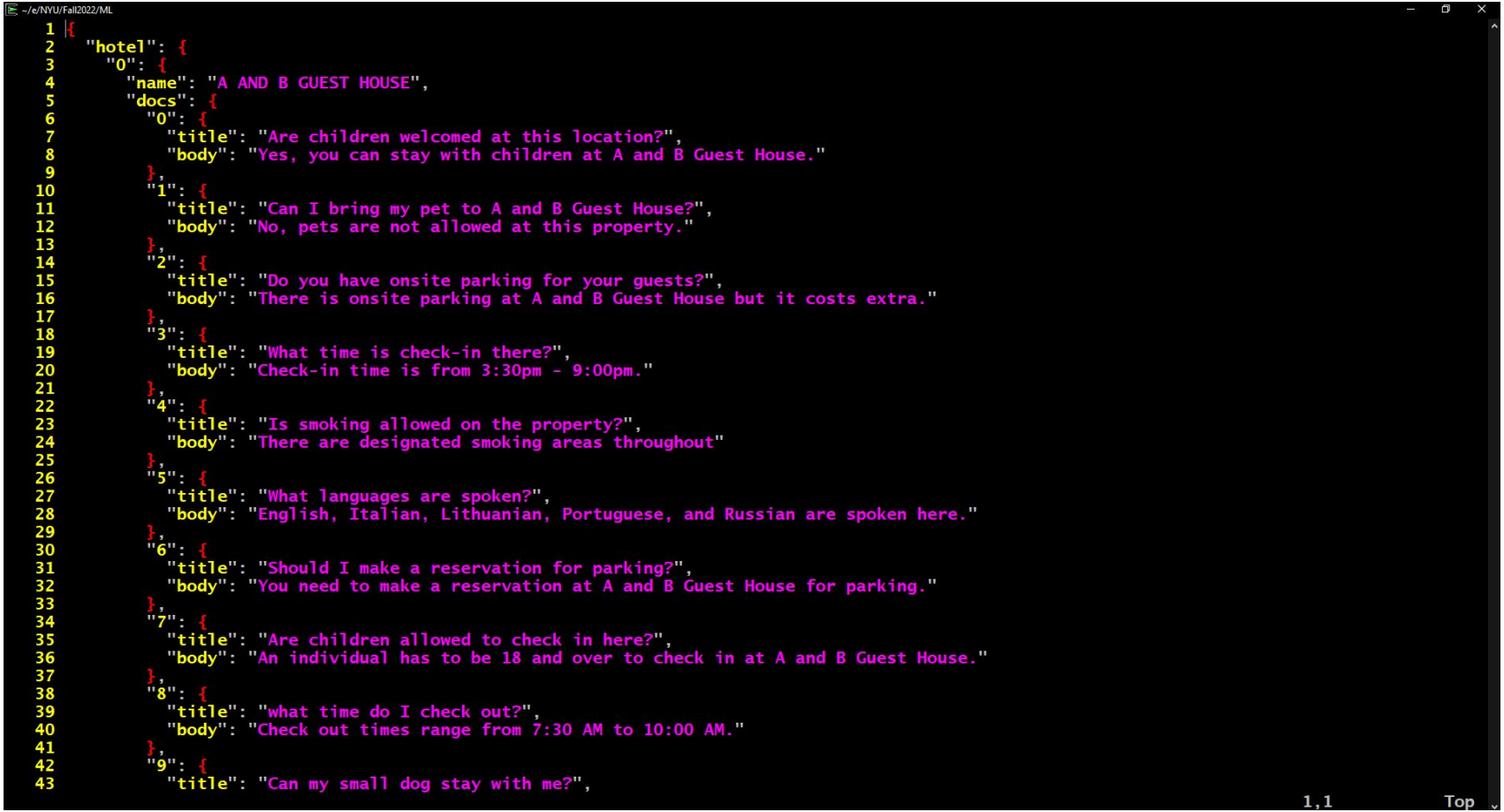
```
"title": "Are there any accessible parking available?",
"body": "It has street and private lot parking.",
"label": "true"
```

```
"title": "Can you accommodate large groups?",
"body": "Cringe.",
```

```
"title": "Can you accommodate large groups?",
"body": "Cringe.",
"label": "false"
```



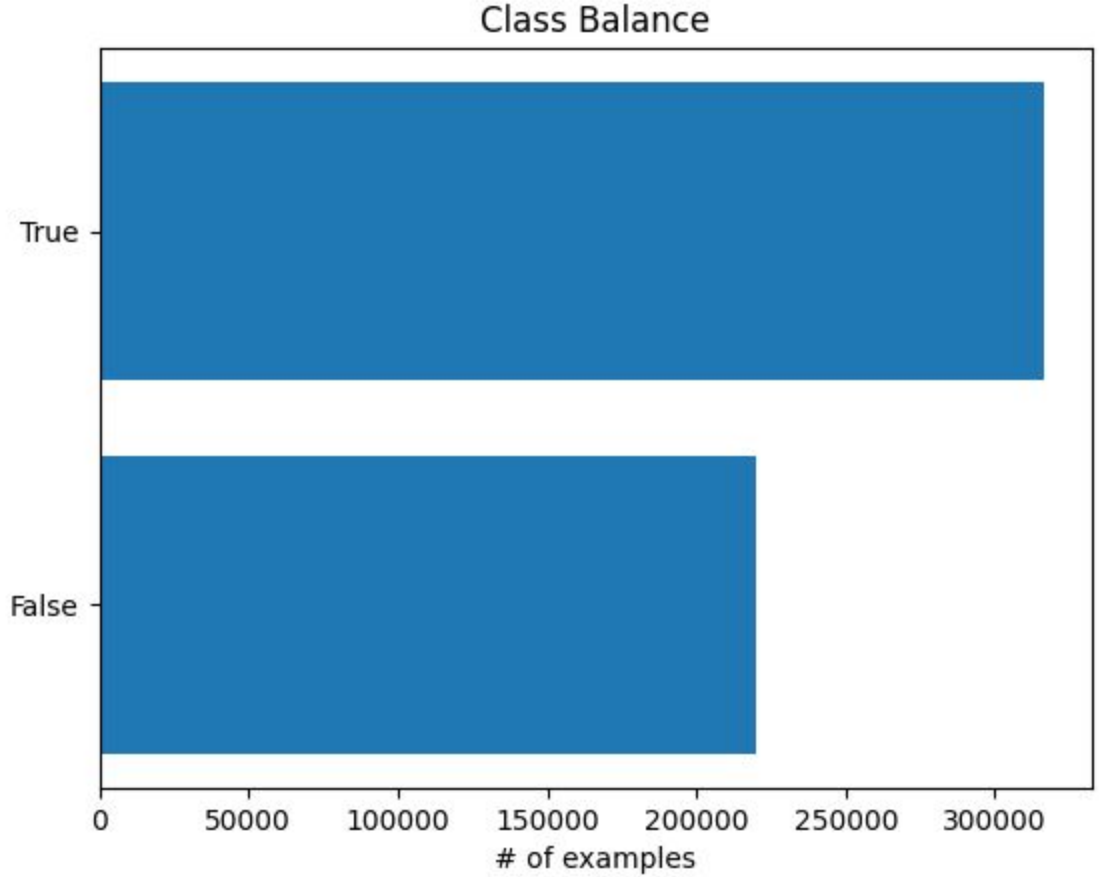
For more info: https://github.com/alexa/alexa-with-dstc10-track2-dataset

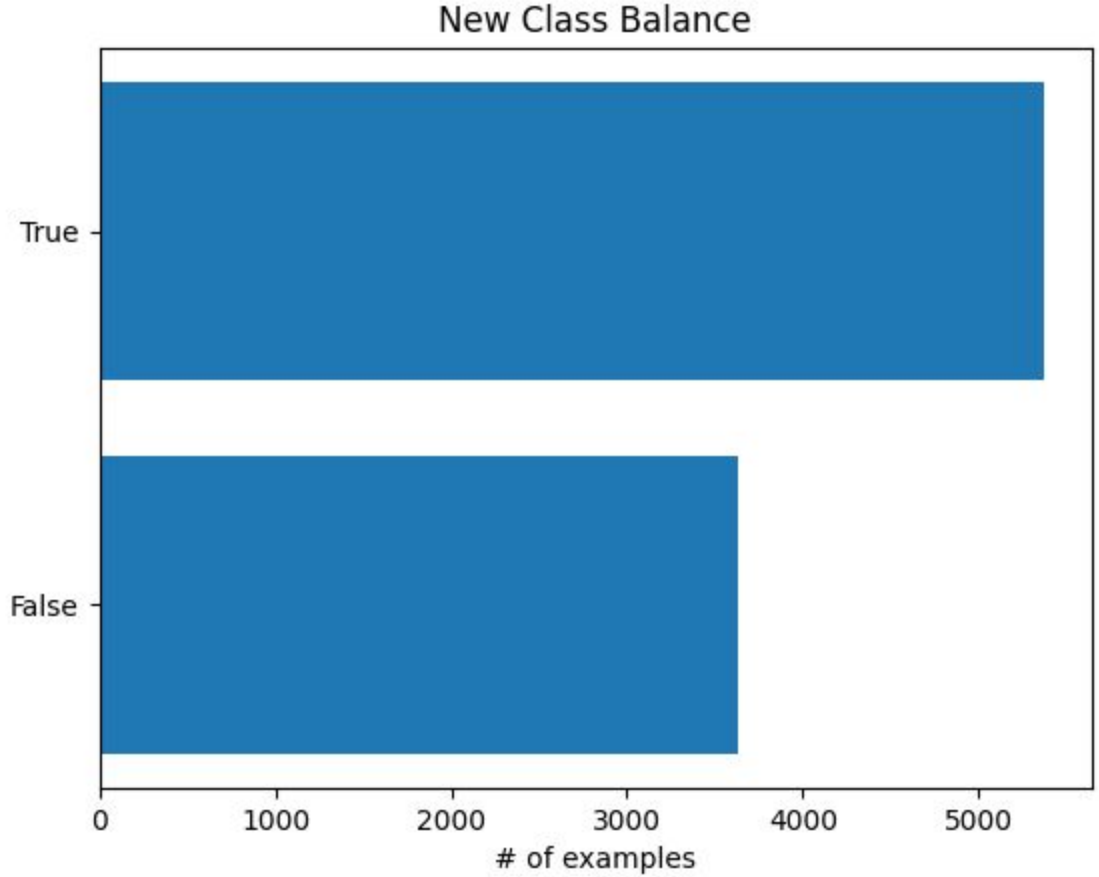

~/e/NYU/Fall2022/ML
{
"hotel": {
"0": {
"name": "A AND B GUEST HOUSE",
"docs": {
"0": {
"title": "Are children welcomed at this location?",
"body": "Yes, you can stay with children at A and B Guest House."
},
"1": {
"title": "Can I bring my pet to A and B Guest House?",
"body": "No, pets are not allowed at this property."
},
"2": {
"title": "Do you have onsite parking for your guests?",
"body": "There is onsite parking at A and B Guest House but it costs extra."
},
"3": {
"title": "What time is check-in there?",
"body": "Check-in time is from 3:30pm - 9:00pm."
},
"4": {
"title": "Is smoking allowed on the property?",
"body": "There are designated smoking areas throughout"
},
"5": {
"title": "What languages are spoken?",
"body": "English, Italian, Lithuanian, Portuguese, and Russian are spoken here."
},
"6": {
"title": "Should I make a reservation for parking?",
"body": "You need to make a reservation at A and B Guest House for parking."
},
"7": {
"title": "Are children allowed to check in here?",
"body": "An individual has to be 18 and over to check in at A and B Guest House."
},
"8": {
"title": "what time do I check out?",
"body": "Check out times range from 7:30 AM to 10:00 AM."
},
"9": {
"title": "Can my small dog stay with me?",
1,1
Top

~/e/NYU/Fall2022/ML

```
[
    {
        "title": "What parking is offered?",
        "body": "It offers off street, street, and validated parking.",
        "label": "true"
    },
    {
        "title": "Can you accommodate large groups?",
        "body": "It does not offer free WiFi.",
        "label": "false"
    },
    {
        "title": "Is there a gym on site?",
        "body": "It does not have an onsite fitness center.",
        "label": "true"
    },
    {
        "title": "What are my payment options available?",
        "body": "The Club Quarters Hotel accepts charge or cash.",
        "label": "true"
    },
    {
        "title": "Do they offer seating outside?",
        "body": "It does not have outdoor seating",
        "label": "true"
    },
    {
        "title": "Do you have wifi availability?",
        "body": "It does offer its guests free WiFi.",
        "label": "true"
    },
    {
        "title": "do you have take out?",
        "body": "Yes, it has take-out.",
        "label": "true"
    },
    {
        "title": "Does it have free WiFi?",
        "body": "No, it is not recommended for groups.",
        "label": "false"
    },
    {
        "title": "Does  it accept credit cards?",
```

"classifier_examples_train.json" [noeol] 2414087L, 70434466B 1,1 Top

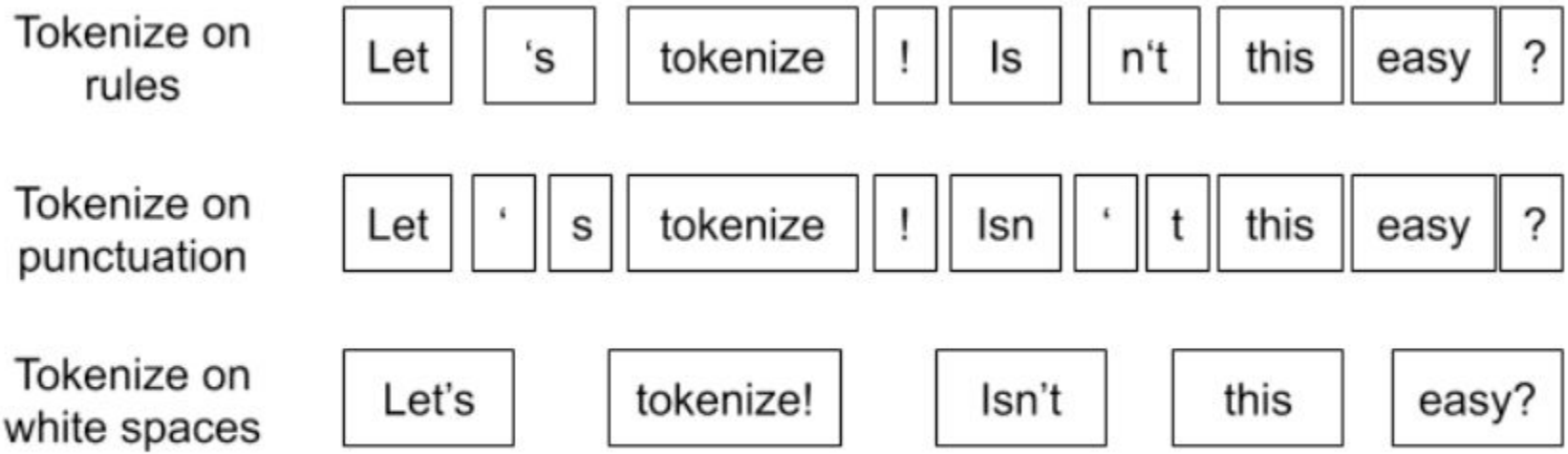

Class Balance
True
False
0
50000
100000
150000
200000
250000
300000
# of examples


New Class Balance
True
False
0
1000
2000
3000
4000
5000
# of examples

```
"title": "Are there any accessible parking available?",
"body": "It has street and private lot parking.",
"label": "true"
```

Tokenization

- Binary
- Count
- Tfidf (term frequency and inverse document frequency)



Let’s tokenize! Isn’t this easy?

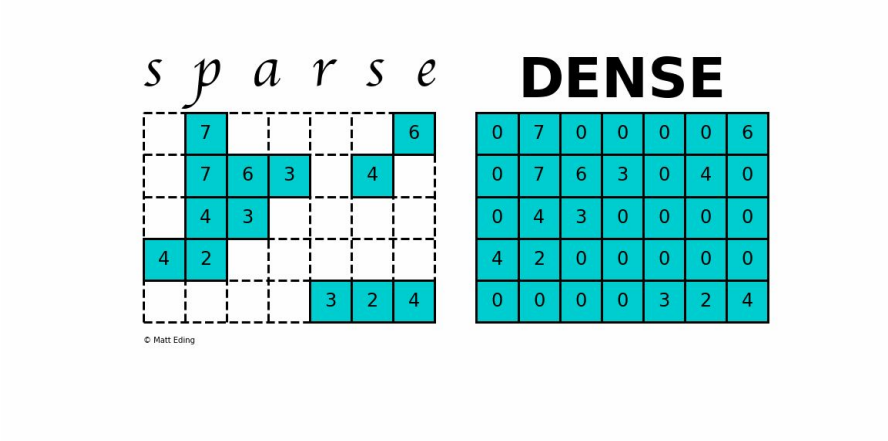
```
# vocab of the tokenizer
tokenizer.vocabulary_
```

```
'galleria': 675,
'joie': 829,
'vivre': 1578,
'thing': 1464,
'connect': 414,
'hampton': 722,
'downtwon': 535,
'convention': 426,
'drop': 545,
'enough': 575,
'instabul': 803,
'multilingual': 999,
'stated': 1390,
'refundable': 1219,
'ride': 1254,
'flyer': 641,
'pregnant': 1165,
'unaccompanied': 1526,
'40': 43,
'inches': 786,
'taller': 1439,
'noon': 1028,
'moderate': 983,
'ammenities': 159,
'opening': 1071,
```

```
0.1981168824235872
0.10654507678857493
0.09724421727884164
0.3715482822964967
0.2837135664094483
0.2731685317029906
0.3707904034826646
0.5444604997328737
0.42113172129544985
0.1884254849966187
```

Our data array/matrix

- Sparse
- Dense




```
(0, 1604)       0.1884254849966187
(0, 1550)       0.42113172129544985
(0, 1402)       0.5444604997328737
(0, 1106)       0.3707904034826646
(0, 1051)       0.2731685317029906
(0, 1047)       0.2837135664094483
(0, 1043)       0.3715482822964967
(0, 815)        0.09724421727884164
(0, 812)        0.10654507678857493
(0, 165)        0.1981168824235872
(1, 1642)       0.17123946985780167
(1, 1617)       0.21902232310438233
(1, 1044)       0.21701672668300245
(1, 1035)       0.1869041190445836
(1, 858)        0.5486673404809553
(1, 815)        0.10677167565523445
(1, 706)        0.36946960039117654
(1, 657)        0.2110448645153775
(1, 520)        0.16121302975125035
(1, 315)        0.23227911491803716
(1, 95)         0.5169621520922558
(2, 1459)       0.16334847919408646
(2, 1336)       0.3160262922127847
(2, 1067)       0.31181700763895026
(2, 1060)       0.3081102636988631
:       :
```

Logistic Regression

Unscaled binary (best @ C = 5)

	precision	recall	f1-score	support
0	0.92	0.77	0.84	443
1	0.84	0.95	0.89	557
accuracy			0.87	1000
macro avg	0.88	0.86	0.86	1000
weighted avg	0.88	0.87	0.87	1000

Logistic Regression

Scaled binary

	precision	recall	f1-score	support
0	0.89	0.75	0.81	443
1	0.82	0.93	0.87	557
accuracy			0.85	1000
macro avg	0.86	0.84	0.84	1000
weighted avg	0.85	0.85	0.85	1000

Logistic Regression

count

	precision	recall	f1-score	support
0	0.71	0.51	0.59	443
1	0.68	0.83	0.75	557
accuracy			0.69	1000
macro avg	0.69	0.67	0.67	1000
weighted avg	0.69	0.69	0.68	1000

	precision	recall	f1-score	support
0	0.70	0.51	0.59	443
1	0.68	0.82	0.74	557
accuracy			0.69	1000
macro avg	0.69	0.67	0.67	1000
weighted avg	0.69	0.69	0.68	1000

Logistic Regression

Tfidf

	precision	recall	f1-score	support
0	0.82	0.63	0.71	443
1	0.75	0.89	0.81	557
accuracy			0.77	1000
macro avg	0.78	0.76	0.76	1000
weighted avg	0.78	0.77	0.77	1000

	precision	recall	f1-score	support
0	0.85	0.70	0.77	443
1	0.79	0.90	0.84	557
accuracy			0.81	1000
macro avg	0.82	0.80	0.81	1000
weighted avg	0.82	0.81	0.81	1000

Good??

```
[1  ]  1  log_model.predict(tokenizer.transform(["Does it offer happy hours? It does not allow children below 6."]))

       array([0])

[159]  1  log_model.predict(tokenizer.transform(["Does it offer a gym? No, it does not have happy hours."]))

       array([1])

[160]  1  log_model.predict(tokenizer.transform(["Does it offer a gym on site? No, it does not have happy hours."]))

       array([0])

[142]  1  log_model.predict(tokenizer.transform(["Is there a gym? No, it does not have happy hours."]))

       array([1])
```

SVM

Dataset problem

- Nystrom
- LinearSVC

```
feature_map_nystroem = Nystroem(gamma=.2, random_state=1, n_components=300)
X_train_nystroem = feature_map_nystroem.fit_transform(X_train_tokenized)
X_test_nystroem = feature_map_nystroem.transform(X_test_tokenized)
```

```
1  support_vec_machine.fit(X_train_nystroem, Y_train)
```

```
LinearSVC()
```

```
[ ]  1  support_vec_machine.score(X_train_nystroem,Y_train)
```

```
0.774442490633097
```

```
[ ]  1  support_vec_machine.score(X_test_nystroem,Y_test)
```

```
0.7742236140625582
```

SVM

Unscaled Binary

	precision	recall	f1-score	support
0	0.92	0.76	0.83	443
1	0.83	0.95	0.89	557
accuracy			0.86	1000
macro avg	0.88	0.85	0.86	1000
weighted avg	0.87	0.86	0.86	1000

	precision	recall	f1-score	support
0	0.85	0.27	0.41	443
1	0.62	0.96	0.76	557
accuracy			0.65	1000
macro avg	0.74	0.61	0.58	1000
weighted avg	0.72	0.65	0.60	1000

	precision	recall	f1-score	support
0	0.00	0.00	0.00	443
1	0.56	1.00	0.72	557
accuracy			0.56	1000
macro avg	0.28	0.50	0.36	1000
weighted avg	0.31	0.56	0.40	1000

SVM

Scaled Tfidf

	precision	recall	f1-score	support
0	0.87	0.71	0.78	443
1	0.80	0.92	0.85	557
accuracy			0.82	1000
macro avg	0.83	0.81	0.82	1000
weighted avg	0.83	0.82	0.82	1000

	precision	recall	f1-score	support
0	0.88	0.73	0.80	443
1	0.81	0.92	0.86	557
accuracy			0.84	1000
macro avg	0.85	0.83	0.83	1000
weighted avg	0.84	0.84	0.83	1000

	precision	recall	f1-score	support
0	0.77	0.26	0.39	443
1	0.62	0.94	0.74	557
accuracy			0.64	1000
macro avg	0.69	0.60	0.57	1000
weighted avg	0.69	0.64	0.59	1000

Good??

```
1  support_vec_machine.predict(scipy.sparse.csr_matrix.todense(tokenizer.transform(["Does it offer happy hours? It does not allow children below 6."])))
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:593: FutureWarning: np.matrix usage is deprecated in 1.0 and will raise a TypeError in 1.2. Please conv
  warnings.warn(
array([0])
```

```
1  support_vec_machine.predict(scipy.sparse.csr_matrix.todense(tokenizer.transform(["Does it offer a gym? No, it does not have happy hours."])))
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:593: FutureWarning: np.matrix usage is deprecated in 1.0 and will raise a TypeError in 1.2. Please conv
  warnings.warn(
array([1])
```

```
1  support_vec_machine.predict(scipy.sparse.csr_matrix.todense(tokenizer.transform(["Does it offer a gym on site? No, it does not have happy hours."])))
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:593: FutureWarning: np.matrix usage is deprecated in 1.0 and will raise a TypeError in 1.2. Please conv
  warnings.warn(
array([0])
```

```
1  support_vec_machine.predict(scipy.sparse.csr_matrix.todense(tokenizer.transform(["Is there a gym? No, it does not have happy hours."])))
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:593: FutureWarning: np.matrix usage is deprecated in 1.0 and will raise a TypeError in 1.2. Please conv
  warnings.warn(
array([1])
```

Neural Network

3 Layer

- Relu
- Relu
- softmax

10 epoch

L2 and dropout regularization

Neural Network

Unscaled Binary

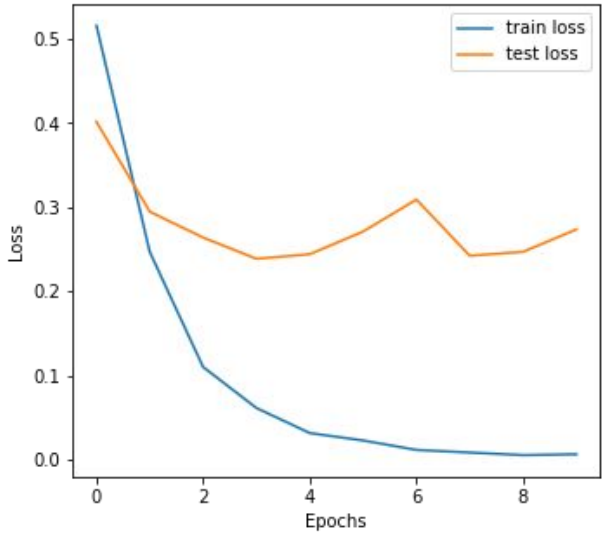
No regularization

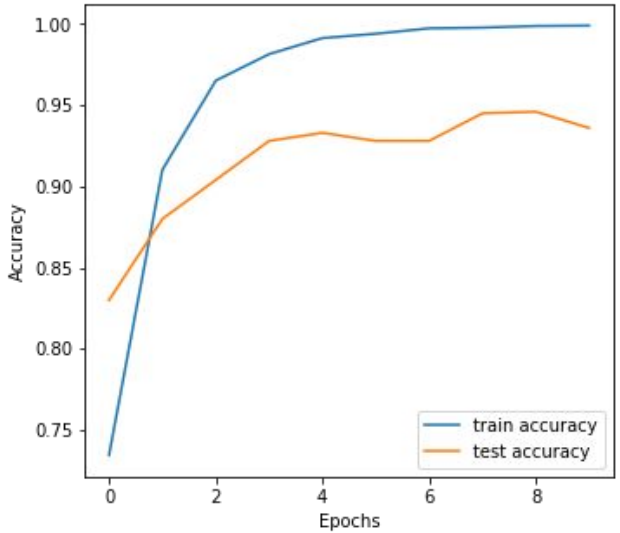
```
Epoch 1/10
282/282 - 3s - loss: 0.5161 - accuracy: 0.7346 - val_loss: 0.4017 - val_accuracy: 0.8300 - 3s/epoch - 10ms/step
Epoch 2/10
282/282 - 2s - loss: 0.2468 - accuracy: 0.9102 - val_loss: 0.2947 - val_accuracy: 0.8800 - 2s/epoch - 6ms/step
Epoch 3/10
282/282 - 2s - loss: 0.1100 - accuracy: 0.9651 - val_loss: 0.2640 - val_accuracy: 0.9040 - 2s/epoch - 6ms/step
Epoch 4/10
282/282 - 2s - loss: 0.0612 - accuracy: 0.9814 - val_loss: 0.2387 - val_accuracy: 0.9280 - 2s/epoch - 6ms/step
Epoch 5/10
282/282 - 2s - loss: 0.0314 - accuracy: 0.9913 - val_loss: 0.2442 - val_accuracy: 0.9330 - 2s/epoch - 6ms/step
Epoch 6/10
282/282 - 2s - loss: 0.0226 - accuracy: 0.9940 - val_loss: 0.2711 - val_accuracy: 0.9280 - 2s/epoch - 6ms/step
Epoch 7/10
282/282 - 2s - loss: 0.0113 - accuracy: 0.9973 - val_loss: 0.3092 - val_accuracy: 0.9280 - 2s/epoch - 6ms/step
Epoch 8/10
282/282 - 1s - loss: 0.0082 - accuracy: 0.9978 - val_loss: 0.2423 - val_accuracy: 0.9450 - 1s/epoch - 5ms/step
Epoch 9/10
282/282 - 1s - loss: 0.0051 - accuracy: 0.9988 - val_loss: 0.2470 - val_accuracy: 0.9460 - 1s/epoch - 5ms/step
Epoch 10/10
282/282 - 2s - loss: 0.0061 - accuracy: 0.9990 - val_loss: 0.2737 - val_accuracy: 0.9360 - 2s/epoch - 6ms/step
```

Neural Network

Unscaled Binary

No regularization





Neural Network

Unscaled Binary

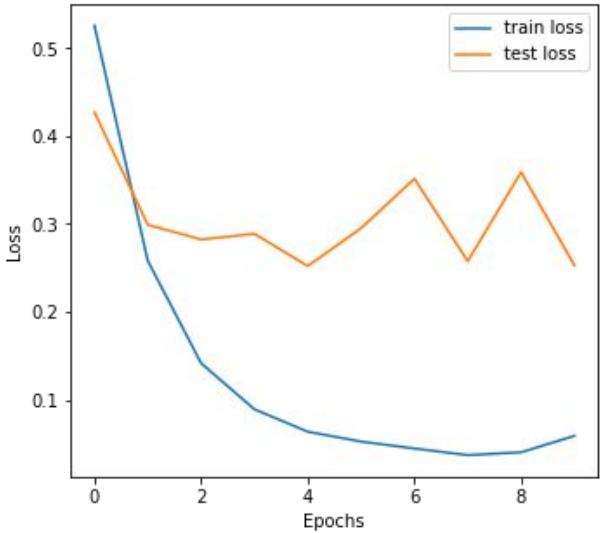
L2 regularization

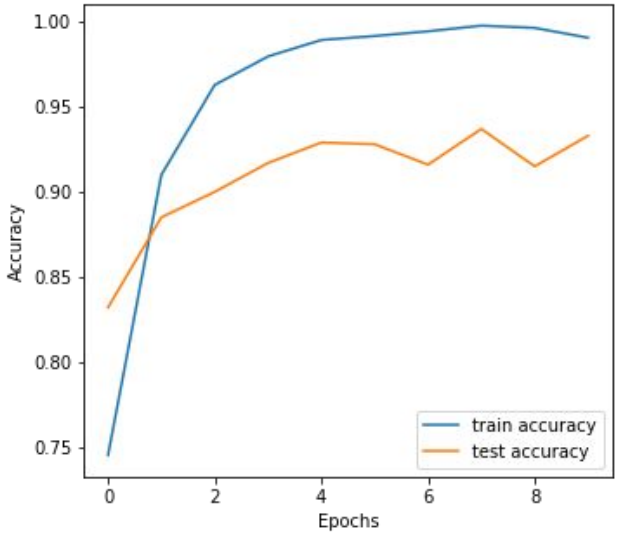
```
Epoch 1/10
282/282 - 3s - loss: 0.5256 - accuracy: 0.7451 - val_loss: 0.4270 - val_accuracy: 0.8320 - 3s/epoch - 10ms/step
Epoch 2/10
282/282 - 2s - loss: 0.2582 - accuracy: 0.9100 - val_loss: 0.2991 - val_accuracy: 0.8850 - 2s/epoch - 7ms/step
Epoch 3/10
282/282 - 2s - loss: 0.1416 - accuracy: 0.9629 - val_loss: 0.2824 - val_accuracy: 0.9000 - 2s/epoch - 7ms/step
Epoch 4/10
282/282 - 1s - loss: 0.0892 - accuracy: 0.9797 - val_loss: 0.2888 - val_accuracy: 0.9170 - 1s/epoch - 5ms/step
Epoch 5/10
282/282 - 1s - loss: 0.0638 - accuracy: 0.9893 - val_loss: 0.2522 - val_accuracy: 0.9290 - 1s/epoch - 5ms/step
Epoch 6/10
282/282 - 2s - loss: 0.0525 - accuracy: 0.9917 - val_loss: 0.2952 - val_accuracy: 0.9280 - 2s/epoch - 6ms/step
Epoch 7/10
282/282 - 2s - loss: 0.0446 - accuracy: 0.9944 - val_loss: 0.3515 - val_accuracy: 0.9160 - 2s/epoch - 7ms/step
Epoch 8/10
282/282 - 2s - loss: 0.0371 - accuracy: 0.9978 - val_loss: 0.2577 - val_accuracy: 0.9370 - 2s/epoch - 7ms/step
Epoch 9/10
282/282 - 2s - loss: 0.0403 - accuracy: 0.9964 - val_loss: 0.3589 - val_accuracy: 0.9150 - 2s/epoch - 6ms/step
Epoch 10/10
282/282 - 2s - loss: 0.0590 - accuracy: 0.9907 - val_loss: 0.2530 - val_accuracy: 0.9330 - 2s/epoch - 7ms/step
```

Neural Network

Unscaled Binary

L2 regularization





Neural Network

Unscaled Binary

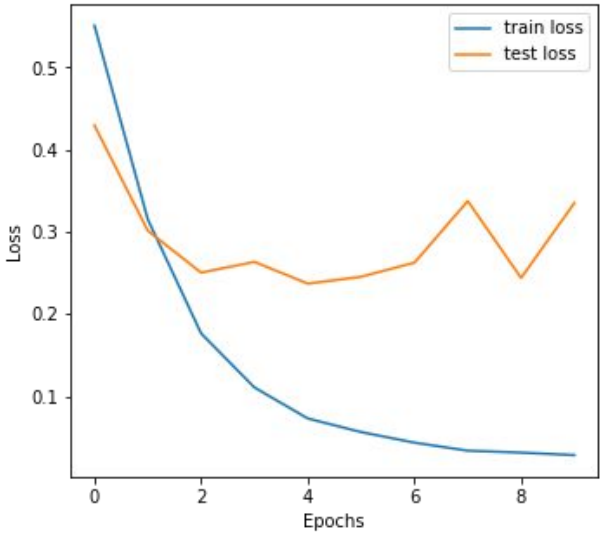
Dropout regularization

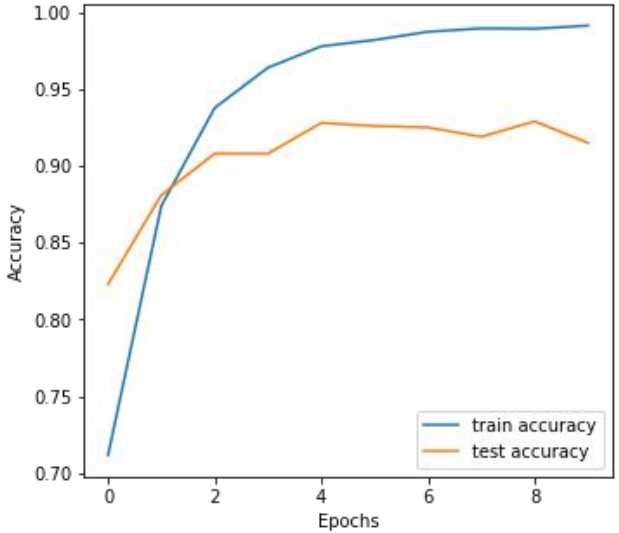
```
Epoch 1/10
282/282 - 2s - loss: 0.5509 - accuracy: 0.7117 - val_loss: 0.4295 - val_accuracy: 0.8230 - 2s/epoch - 9ms/step
Epoch 2/10
282/282 - 2s - loss: 0.3153 - accuracy: 0.8740 - val_loss: 0.3015 - val_accuracy: 0.8810 - 2s/epoch - 5ms/step
Epoch 3/10
282/282 - 2s - loss: 0.1764 - accuracy: 0.9378 - val_loss: 0.2504 - val_accuracy: 0.9080 - 2s/epoch - 5ms/step
Epoch 4/10
282/282 - 2s - loss: 0.1107 - accuracy: 0.9640 - val_loss: 0.2635 - val_accuracy: 0.9080 - 2s/epoch - 6ms/step
Epoch 5/10
282/282 - 2s - loss: 0.0730 - accuracy: 0.9779 - val_loss: 0.2370 - val_accuracy: 0.9280 - 2s/epoch - 6ms/step
Epoch 6/10
282/282 - 2s - loss: 0.0564 - accuracy: 0.9820 - val_loss: 0.2454 - val_accuracy: 0.9260 - 2s/epoch - 6ms/step
Epoch 7/10
282/282 - 1s - loss: 0.0436 - accuracy: 0.9873 - val_loss: 0.2625 - val_accuracy: 0.9250 - 1s/epoch - 5ms/step
Epoch 8/10
282/282 - 2s - loss: 0.0338 - accuracy: 0.9896 - val_loss: 0.3375 - val_accuracy: 0.9190 - 2s/epoch - 6ms/step
Epoch 9/10
282/282 - 2s - loss: 0.0314 - accuracy: 0.9893 - val_loss: 0.2440 - val_accuracy: 0.9290 - 2s/epoch - 5ms/step
Epoch 10/10
282/282 - 1s - loss: 0.0284 - accuracy: 0.9913 - val_loss: 0.3352 - val_accuracy: 0.9150 - 1s/epoch - 5ms/step
```

Neural Network

Unscaled Binary

Dropout regularization





Neural Network

Scaled tfidf

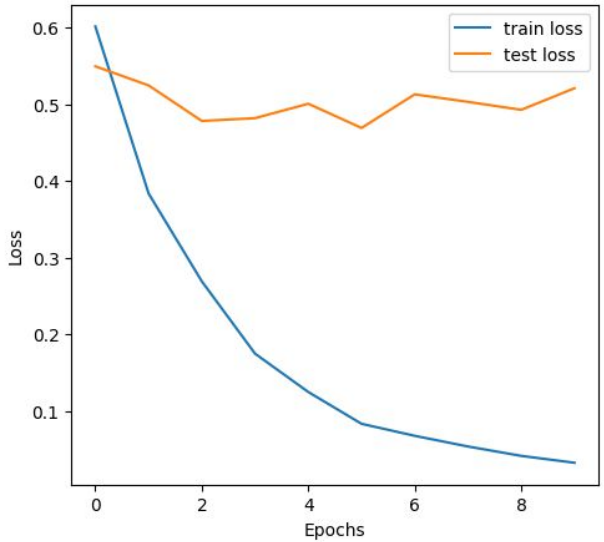
No regularization

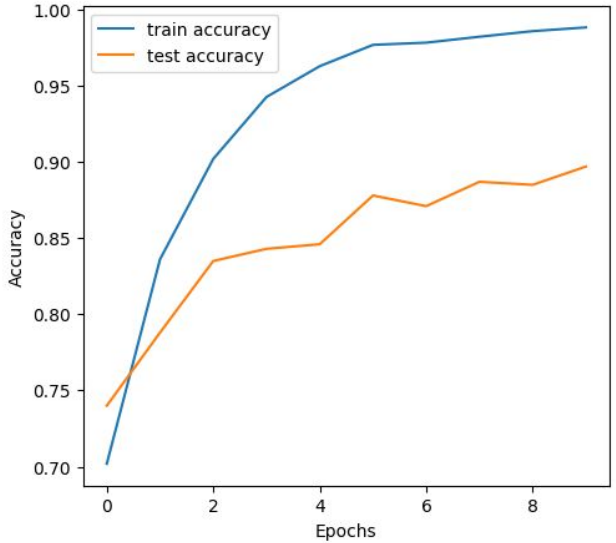
```
Epoch 1/10
282/282 - 2s - loss: 0.6013 - accuracy: 0.7019 - val_loss: 0.5493 - val_accuracy: 0.7400 - 2s/epoch - 9ms/step
Epoch 2/10
282/282 - 1s - loss: 0.3836 - accuracy: 0.8361 - val_loss: 0.5245 - val_accuracy: 0.7880 - 1s/epoch - 4ms/step
Epoch 3/10
282/282 - 1s - loss: 0.2690 - accuracy: 0.9020 - val_loss: 0.4781 - val_accuracy: 0.8350 - 1s/epoch - 4ms/step
Epoch 4/10
282/282 - 1s - loss: 0.1748 - accuracy: 0.9427 - val_loss: 0.4817 - val_accuracy: 0.8430 - 1s/epoch - 4ms/step
Epoch 5/10
282/282 - 1s - loss: 0.1248 - accuracy: 0.9629 - val_loss: 0.5005 - val_accuracy: 0.8460 - 1s/epoch - 4ms/step
Epoch 6/10
282/282 - 1s - loss: 0.0834 - accuracy: 0.9769 - val_loss: 0.4690 - val_accuracy: 0.8780 - 1s/epoch - 4ms/step
Epoch 7/10
282/282 - 1s - loss: 0.0677 - accuracy: 0.9783 - val_loss: 0.5128 - val_accuracy: 0.8710 - 1s/epoch - 4ms/step
Epoch 8/10
282/282 - 1s - loss: 0.0538 - accuracy: 0.9822 - val_loss: 0.5030 - val_accuracy: 0.8870 - 1s/epoch - 4ms/step
Epoch 9/10
282/282 - 1s - loss: 0.0415 - accuracy: 0.9859 - val_loss: 0.4926 - val_accuracy: 0.8850 - 994ms/epoch - 4ms/step
Epoch 10/10
282/282 - 1s - loss: 0.0326 - accuracy: 0.9883 - val_loss: 0.5207 - val_accuracy: 0.8970 - 1s/epoch - 4ms/step
```

Neural Network

Scaled tfidf

No regularization





Neural Network

Scaled tfidf

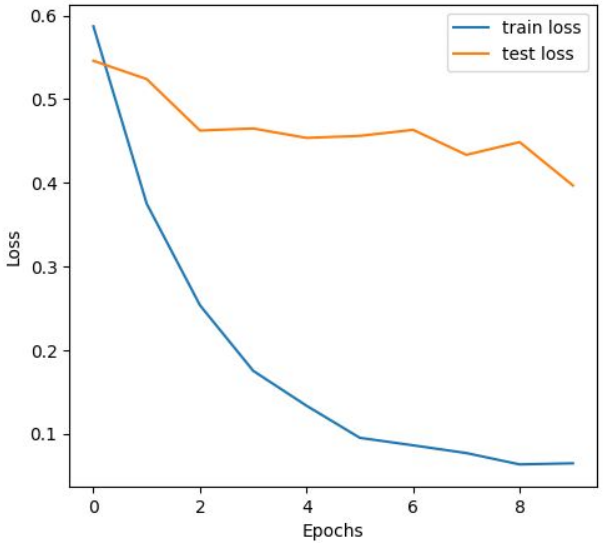
L2 regularization

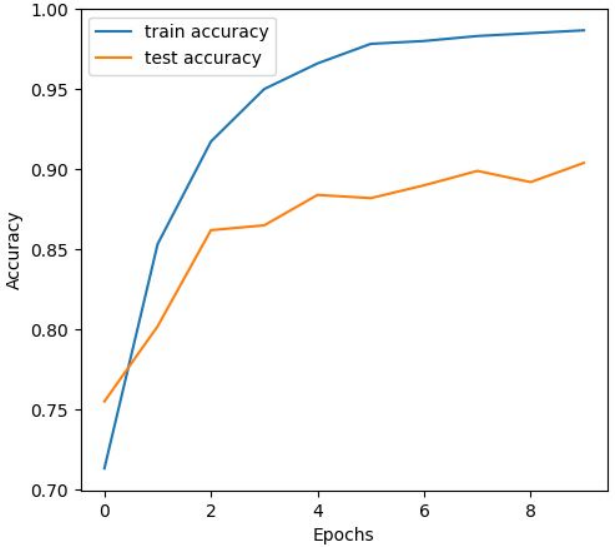
```
Epoch 1/10
282/282 - 2s - loss: 0.5871 - accuracy: 0.7132 - val_loss: 0.5459 - val_accuracy: 0.7550 - 2s/epoch - 8ms/step
Epoch 2/10
282/282 - 1s - loss: 0.3750 - accuracy: 0.8533 - val_loss: 0.5241 - val_accuracy: 0.8020 - 1s/epoch - 5ms/step
Epoch 3/10
282/282 - 1s - loss: 0.2535 - accuracy: 0.9173 - val_loss: 0.4625 - val_accuracy: 0.8620 - 1s/epoch - 5ms/step
Epoch 4/10
282/282 - 1s - loss: 0.1751 - accuracy: 0.9501 - val_loss: 0.4650 - val_accuracy: 0.8650 - 1s/epoch - 5ms/step
Epoch 5/10
282/282 - 1s - loss: 0.1334 - accuracy: 0.9661 - val_loss: 0.4537 - val_accuracy: 0.8840 - 1s/epoch - 5ms/step
Epoch 6/10
282/282 - 1s - loss: 0.0950 - accuracy: 0.9783 - val_loss: 0.4561 - val_accuracy: 0.8820 - 1s/epoch - 5ms/step
Epoch 7/10
282/282 - 1s - loss: 0.0861 - accuracy: 0.9801 - val_loss: 0.4634 - val_accuracy: 0.8900 - 1s/epoch - 5ms/step
Epoch 8/10
282/282 - 1s - loss: 0.0768 - accuracy: 0.9832 - val_loss: 0.4335 - val_accuracy: 0.8990 - 1s/epoch - 5ms/step
Epoch 9/10
282/282 - 1s - loss: 0.0633 - accuracy: 0.9850 - val_loss: 0.4487 - val_accuracy: 0.8920 - 1s/epoch - 5ms/step
Epoch 10/10
282/282 - 1s - loss: 0.0646 - accuracy: 0.9868 - val_loss: 0.3968 - val_accuracy: 0.9040 - 1s/epoch - 4ms/step
```

Neural Network

Scaled tfidf

L2 regularization





Neural Network

Scaled tfidf

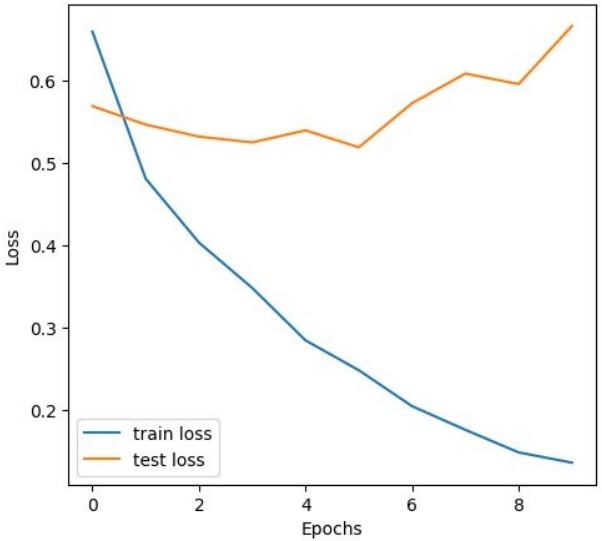
Dropout regularization

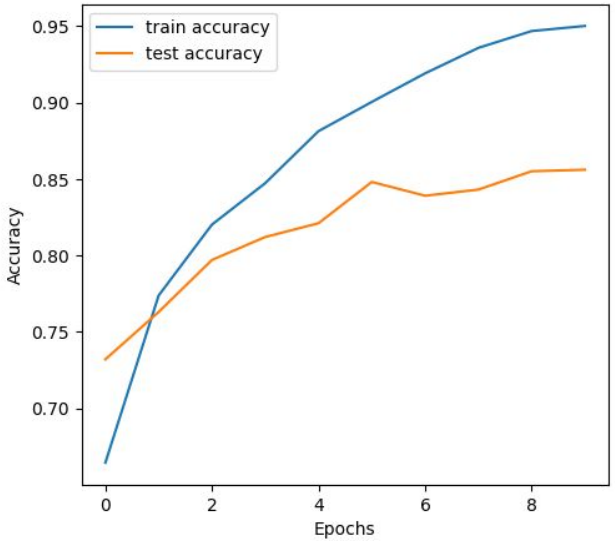
```
Epoch 1/10
282/282 - 2s - loss: 0.6602 - accuracy: 0.6644 - val_loss: 0.5696 - val_accuracy: 0.7320 - 2s/epoch - 7ms/step
Epoch 2/10
282/282 - 1s - loss: 0.4813 - accuracy: 0.7738 - val_loss: 0.5471 - val_accuracy: 0.7630 - 1s/epoch - 4ms/step
Epoch 3/10
282/282 - 1s - loss: 0.4036 - accuracy: 0.8201 - val_loss: 0.5325 - val_accuracy: 0.7970 - 1s/epoch - 4ms/step
Epoch 4/10
282/282 - 1s - loss: 0.3483 - accuracy: 0.8472 - val_loss: 0.5255 - val_accuracy: 0.8120 - 1s/epoch - 4ms/step
Epoch 5/10
282/282 - 1s - loss: 0.2846 - accuracy: 0.8812 - val_loss: 0.5401 - val_accuracy: 0.8210 - 1s/epoch - 4ms/step
Epoch 6/10
282/282 - 1s - loss: 0.2482 - accuracy: 0.9003 - val_loss: 0.5195 - val_accuracy: 0.8480 - 1s/epoch - 4ms/step
Epoch 7/10
282/282 - 1s - loss: 0.2044 - accuracy: 0.9190 - val_loss: 0.5732 - val_accuracy: 0.8390 - 1s/epoch - 4ms/step
Epoch 8/10
282/282 - 1s - loss: 0.1757 - accuracy: 0.9357 - val_loss: 0.6093 - val_accuracy: 0.8430 - 1s/epoch - 4ms/step
Epoch 9/10
282/282 - 1s - loss: 0.1482 - accuracy: 0.9467 - val_loss: 0.5965 - val_accuracy: 0.8550 - 1s/epoch - 4ms/step
Epoch 10/10
282/282 - 1s - loss: 0.1357 - accuracy: 0.9500 - val_loss: 0.6672 - val_accuracy: 0.8560 - 1s/epoch - 4ms/step
```

Neural Network

Scaled tfidf

Dropout regularization





Good??

```
1  model.predict(scipy.sparse.csr_matrix.todense(tokenizer.transform(["Does it offer happy hours? It does not allow children below 6."])))
```

```
1/1 [==============================] - 0s 108ms/step
array([[0.07494579, 0.9250542 ]], dtype=float32)
```

```
1  model.predict(scipy.sparse.csr_matrix.todense(tokenizer.transform(["Does it offer a gym? No, it does not have happy hours."])))
```

```
1/1 [==============================] - 0s 55ms/step
array([[0.02371382, 0.9762862 ]], dtype=float32)
```

```
1  model.predict(scipy.sparse.csr_matrix.todense(tokenizer.transform(["Does it offer a gym on site? No, it does not have happy hours."])))
```

```
1/1 [==============================] - 0s 19ms/step
array([[0.05099352, 0.9490065 ]], dtype=float32)
```

```
1  model.predict(scipy.sparse.csr_matrix.todense(tokenizer.transform(["Is there a gym? No, it does not have happy hours."])))
```

```
1/1 [==============================] - 0s 19ms/step
array([[9.1008429e-04, 9.9908996e-01]], dtype=float32)
```

BERT Transformer (MLM)

```
[261]  1  #Checks for dict weight
       2  state_dict = model.state_dict()
       3  state_dict["bert.embeddings.word_embeddings.weight"]
```

```
tensor([[-0.0102, -0.0615, -0.0265,  ..., -0.0199, -0.0372, -0.0098],
        [-0.0117, -0.0600, -0.0323,  ..., -0.0168, -0.0401, -0.0107],
        [-0.0198, -0.0627, -0.0326,  ..., -0.0165, -0.0420, -0.0032],
        ...,
        [-0.0218, -0.0556, -0.0135,  ..., -0.0043, -0.0151, -0.0249],
        [-0.0462, -0.0565, -0.0019,  ...,  0.0157, -0.0139, -0.0095],
        [ 0.0015, -0.0821, -0.0160,  ..., -0.0081, -0.0475,  0.0753]])
```

BERT Transformer (MLM)

```
[ ]  1  sentence = "Does it offer happy hours?"
     2  body = "It does not allow children below 6."
     3  pair = (sentence, body)
     4  _single_test(pair)
```

```
False
```

```
[267]  1  sentence = "Does it offer a gym?"
       2  body = " No, it does not have happy hours."
       3  pair = (sentence, body)
       4  _single_test(pair)
```

```
False
```

```
[268]  1  sentence = "Does it offer a gym on site?"
       2  body = " No, it does not have happy hours."
       3  pair = (sentence, body)
       4  _single_test(pair)
```

```
False
```

```
[270]  1  sentence = "Is there a gym?"
       2  body = "No, it does not have happy hours."
       3  pair = (sentence, body)
       4  _single_test(pair)
```

```
False
```

```
[271]  1  sentence = "Is there a gym?"
       2  body = "Yes, there is a gym."
       3  pair = (sentence, body)
       4  _single_test(pair)
```

```
True
```

Thank you!