

Table of Experiments NER Model 3/2018-6/2018

<u>Tests:</u>	<u>-- Acc(from eval)-- (TRAIN)</u>	<u>F1-- (TRAIN)</u>	<u>Classes Predicted --(TRAIN)</u>	<u>Loss on last epoch/val loss (TRAIN)</u>	<u>Precision/ Recall (TRAIN)</u>
100 epochs, decay=0.9 , batchsize =32, learningra te=0.0001, dropout=0 .5	~83	.13/100	NA	NA but continued to go down	
300 epochs, decay=0.9, batchsize= 32, learningrat e=0.0001, dropout=0. 5	~83	0	0s, 6s and 9s	Did not look as before creation of table, but recorded that loss starting going up at 295 epochs and I remember accuracy stopping improveme nt early on before 100 epochs.	

150 epochs, decay=0.9, batchsize=32, learningrate=0.0001, dropout=0.5	~83	0	0s, 1s (only 1), 8s(only 8) and 9s	Did not look but somewhere around 1.6 or 1.7	
2 epochs, decay=0.9 , learningrate=0.0001, batchsize =32, dropout=0.5	~83	0.04163197 335553705 4/100	0, 1, 2, 4, 5, 6, 9	1.8899/1.9004	
2 epochs, decay=0, learningrate=0.0001, batchsize=32, dropout=0.5:	~83	0	0, 9	0.1041/0.0977	
2 epochs, decay=0.9, learningrate=0.001, batchsize=32, dropout=0.5:	~83	0	0, 9	0.1756/0.9653	

2 epochs, decay=0.9 , learningra te=0.0000 1, batchsize =32, dropout=0 .5:	3.2457	4.4642030 19357229/1 00	0, 1, 2, 3, 4, 5, 6, 7, 8	2.2873/2.2 802	
2 epochs, decay=0.1, learningrat e=0.0001, batchsize= 32, dropout=0. 5:	~83	0	0, 9	0.3089/0.3 149	
2 epochs, decay=0.5, learningrat e=0.0001, batchsize= 32, dropout=0. 5:	~83	0.00831981 363617455 /100	0, 5, 6, 9	1.7557/1.77 66	
NER Paper: 2 epochs, decay=0.0 5, learningra te=0.015, batchsize =32, dropout=0 .5:	85.6	26.394331 418755705 /100	0, 2, 4, 6, 8, 9	0.0745/0.07 45	

NER Paper: 20 epochs, decay=0.05, learningrate=0.015, batchsize =32, dropout=0.5:	88.5	39.685975 864043025	0, 2, 4, 6, 8, 9	0.0568/0.0622	{'precision': 0.49478835642557517} {'recall': 0.33129069322098814}
4)NER Paper: 50 epochs, decay=0.05, learningrate=0.015, batchsize =32, dropout=0.5: **REDO	86.5	33.148	0, 2, 4, 5, 6, 7, 9	Only went to 46 epochs... 0.0701/0.0776	{'precision': 0.5636493665713118} {'recall': 0.23477594791267714}
*NER Paper: 30 epochs, decay=0.05, learningrate=0.015, batchsize =32, dropout=0.5:	88.787998 337985768	41.2831458 382505348	0, 2, 4, 6, 8, 9	0.0535/0.0606	{'precision': 0.512001512001512} {'recall': 0.34584450402144773}

*NER Paper: 40 epochs, decay=0.0 5, learningra te=0.015, batchsize =32, dropout=0 .5:	87.302390 2249768	35.9130951 7375561	0, 2, 4, 6, 8, 9	0.0607/0.0 666	{'precision': 0.4919671 281557711 } 'recall': 0.2827779 90552789 46}
*NER Paper: 2 epochs, decay=0.0 5, learningra te=0.015, batchsize =10 on training, 32 on eval, dropout=0 .5:	85.968048 48222924	29.600194 63141676	0, 2, 4, 6, 8, 9	0.0707/0.07 29	{'precision': 0.4057358 826144953 4} 'recall': 0.2329886 378143751 }
NER Paper: 20 epochs, decay=0.05 ,, learningrat e=0.015, batchsize= 10 on training, 32 on eval, dropout=0. 5:	86.914905 63350538	34.712093 48162748	0, 2, 4, 6, 8, 9	0.0638/0.0 697	{'precision': 0.5129826 89747004} 'recall': 0.2623090 34426996 9}

*NER Paper: 2 epochs, decay=0.0 5, learningra te=0.0105, batchsize =10 on training, 32 on eval, dropout=0 .5: ** <u>h/e</u> <u>precision</u> <u>much</u> <u>better with</u> <u>bachsize=</u> <u>32 so may</u> <u>want to</u> <u>run longer</u> <u>with that</u> <u>and test</u>	86.419868 2847054	31.638517 99451764 5	Forgot to capture.	0.0694/0.0 727	{'precision': 0.5146720 36823935 6} {'recall': 0.2283926 975615983 6}
3)NER Paper: 20 epochs, decay=0.05 ,, learningrat e=0.0105, batchsize= 10 on training, 32 on eval, dropout=0. 5:	87.8789515 8161486	37.4881316 5945775	0, 2, 4, 6, 8, 9	0.0592/0.0 647	{'precision': 0.4929597 003537490 3} {'recall': 0.3024384 016341120 6}

NER Paper: 2 epochs, decay=0.05 ,, learningrate=0.0105, batchsize=32 on training, 32 on eval, dropout=0. 5:	85.702849 90251497	27.6452912 21788888	0, 2, 6, 8, 9	0.0787/0.0 848	<code>{'precision': 0.555569 83168444 04} {'recall': 0.1840078 30120430 65}</code>
**NER Paper: 20 epochs, decay=0.0 5, learningra te=0.0105, batchsize =32 on training, 32 on eval, dropout=0 .5:	88.6706	40.875324 217471224	0, 2, 4, 6, 8, 9	0.0552/0.0 612	<code>{'precision': 0.5006168 270416975 } {'recall': 0.3453763 989957019 4}</code>
NER Paper: 30 epochs, decay=0.05 ,, learningrate=0.0105, batchsize=32 on training, 32 on eval, dropout=0. 5:	87.2100618 3055775	36.552806 143513806	0, 2, 4, 6, 8, 9	0.0620/0.0 683	<code>{'precision': 0.5174563 591022444 } {'recall': 0.2825652 155410869 }</code>

NER Paper: 2 epochs, decay=0.05 ,, learningrate=0.015, batchsize=32 on training, 32 on eval, dropout=0. 68:	84.854705 55591024	20.2972115 54516615	0, 2, 6, 8, 9	0.0806/0.0 870	{'precision': 0.47145516 59944151} 'recall': 0.1293246 521128558 8}
NER Paper: 2 epochs, decay=0.05 ,, learningrate=0.015, batchsize=10 on training, 32 on eval, dropout=0. 68:	85.367913 91850545	26.968443 15444945	0, 2, 6, 8, 9	0.0817/0.08 58	{'precision': 0.49139192 07831664} 'recall': 0.1858376 952210732 4}
NER Paper: 2 epochs, decay=0.05 ,, learningrate=0.0105, batchsize=10 on training, 32 on eval, dropout=0. 68:	85.521630 87304353	27.6590410 52961452	0, 2, 6, 8, 9	0.0798/0.0 852	{'precision': 0.5246700 749019142 } 'recall': 0.18779522 532873738 }

2 epochs, decay=0.05 - learningrate=0.0105, batchsize=20 on training, 32 on eval, dropout=0.5:	86.356024 1821816	30.972359 328726558	0, 2, 4, 6, 8, 9	0.0742/0.0 772	{'precision': 0.5629696 086127621 } {'recall': 0.21362611 174943616 }
2 epochs, decay=0.05, learningrate=0.0105, batchsize=10 on training, 32 on eval, dropout=0.5:	86.883474 69072444	34.893175 33085373	0, 2, 4, 6, 8, 9	0.0671/0.07 02	{'precision': 0.5410407 725321889 } {'recall': 0.2575003 191625175 3}
20 epochs, decay=0.05, learningrate=0.0105, batchsize=10 on training, 32 on eval, dropout=0.5:	91.1173209 0501471	53.014486 08875604 4	0, 2, 4, 6, 8, 9	0.0430/0.0 505	{'precision': 0.5766594 967735481 } {'recall': 0.4905740 669815737 }

50 epochs, decay=0.0 05, learningrat e=0.0105, batchsize= 10 on training, 32 on eval, dropout=0. 5:	92.931966 74213367	62.104577 80471925 6	0, 2, 4, 6, 8, 9	0.0343/0.0 459	{'precision': 0.6573044 387415645 } {'recall': 0.5885782 37371803}
100 epochs, decay=0.0 05, learningrat e=0.0105, batchsize= 10 on training, 32 on eval, dropout=0. 5:	92.472289 20396226	59.625559 62555963	0, 2, 4, 6, 8, 9	0.0357/0.0 471	{'precision': 0.6361267 911419887} {'recall': 0.5610877 05859823 8}
20 epochs, decay=0.0 05, learningrat e=0.0105, batchsize= 20 on training, 32 on eval, dropout=0. 5:					

2 epochs, decay=0.0 005, learningrat e=0.0105, batchsize= 10 on training, 32 on eval, dropout=0. .5:	89.195613 41904813	43.922207 8833571	0, 2, 4, 6, 8, 9	0.0602/0.0 589	{'precision': 0.4692791 485244315 3} 'recall': 0.4127835 227030937 6}
<u>20</u> <u>epochs,</u> <u>decay=0.0</u> <u>005,</u> <u>learningra</u> <u>te=0.0105,</u> <u>batchsize</u> <u>=10 on</u> <u>training,</u> <u>32 on eval,</u> <u>dropout=0</u> .5:	93.96083 90097288 6	65.522323 36367577	0, 2, 3, 4, 6, 8, 9	0.0346/0.0 397	{'precision': 0.6676383 551963253 } 'recall': 0.64326141 53793779}
20 epochs, decay=0.0 005, learningrat e=0.0105, batchsize= 32 on training, 32 on eval, dropout=0. .5:	92.244905 97728132	56.874972 933177425	0, 2, 4, 6, 8, 9	0.0399/0.0 467	{'precision': 0.5789798 527531632 } 'recall': 0.5588748 457381165 }

50 epochs, decay=0.0 005, learningrat e=0.0105, batchsize= 10 on training, 32 on eval, dropout=0. 5: * not sure why got worse than 20 epochs: loss was steadily declining so worth trying again for 75- could just be luck of run.	92.796420 80139082	59.301342 23822018	0, 2, 3, 4, 6, 8, 9	0.0357/0.0 449	{'precision': 0.6045627 376425855 } {'recall': 0.58189710 20043406}
<u>75 epochs,</u> <u>decay=0.0</u> <u>005,</u> <u>learningra</u> <u>te=0.0105,</u> <u>batchsize</u> <u>=10 on</u> <u>training,</u> <u>32 on eval,</u> <u>dropout=0</u> <u>.5:</u>	96.237617 927424	78.45620 64096529 9	0, 2, 3, 4, 6, 8, 9	0.0204/0. 0368	{'precision': 0.7888826 743535688 } {'recall': 0.7802885 229158688 }

2 epochs, decay=0.0 0005, learningrat e=0.0105, batchsize= 10 on training, 32 on eval, dropout=0. 5: 2 epochs, decay=0.0 005, learningrat e=0.01005, batchsize= 10 on training, 32 on eval, dropout=0. 5: 5 epochs, decay=0.0 005, learningrat e=0.01005, batchsize= 10 on training, 32 on eval, dropout=0. 5: *may want to run longer because loss seems to decrease quickly with these settings	86.462103 6140673	32.544395 0399072	0, 2, 4, 6, 8, 9	0.0724/0.0 736	{'precision': 0.4635719 905586152 5} {'recall': 0.2507340 737903740 5}
2 epochs, decay=0.0 005, learningrat e=0.01005, batchsize= 10 on training, 32 on eval, dropout=0. 5: 2 epochs, decay=0.0 005, learningrat e=0.01005, batchsize= 10 on training, 32 on eval, dropout=0. 5: 5 epochs, decay=0.0 005, learningrat e=0.01005, batchsize= 10 on training, 32 on eval, dropout=0. 5: *may want to run longer because loss seems to decrease quickly with these settings	88.020881 9326101	37.247049 46293594	0, 2, 4, 6, 8, 9	0.0639/0.0 644	{'precision': 0.4942981 838659721 3} {'recall': 0.2988212 264351674 5}
2 epochs, decay=0.0 005, learningrat e=0.01005, batchsize= 10 on training, 32 on eval, dropout=0. 5: 2 epochs, decay=0.0 005, learningrat e=0.01005, batchsize= 10 on training, 32 on eval, dropout=0. 5: 5 epochs, decay=0.0 005, learningrat e=0.01005, batchsize= 10 on training, 32 on eval, dropout=0. 5: *may want to run longer because loss seems to decrease quickly with these settings	88.555207 95988626	40.166275 62990921	0, 2, 4, 6, 8, 9	0.0582/0.0 608	{'precision': 0.50117736 90574683} {'recall': 0.3351206 434316354 }

2 epochs, decay=0.0 005, learningrat e=0.00105, batchsize= 10 on training, 32 on eval, dropout=0. 5:	87.650095 02949106	36.738447 39060464	0, 2, 4, 6, 8, 9	0.0632/0.0 659	{'precision': 0.4595475 460122699 4} 'recall': 0.3060130 218307162 }
*2 epochs, decay=0.0 005, learningrat e=0.01000 5, batchsize= 10 on training, 32 on eval, dropout=0. 5:	89.245706 48410527	43.911480 82009087	0, 2, 4, 6, 8, 9	0.0596/0.0 580	{'precision': 0.4683220 829315332 7} 'recall': 0.4133367 377335206 }
20 epochs, decay=0.0 005, learningrat e=0.01000 5, batchsize= 10 on training, 32 on eval, dropout=0. 5:	92.548902 12699082	58.1720711 2970711	0, 2, 3, 4, 6, 8, 9	0.0385/0.0 445	{'precision': 0.5961409 620795927 } 'recall': 0.56798161 62389889}

<u>75 epochs, decay=0.0 005, learningra te=0.015, batchsize =10 on training, 32 on eval, dropout=0 .5 w/ dev(valida tion data) in training: 5/17/18</u>	Train: 96.629 dev: 95.34	Train: 79.728 dev: 74.032			Train: {'precision': 0.8077830 188679245 } {'recall': 0.7870547 682880122 } Dev: {'precision': 0.7550306 211723534 } {'recall': 0.7261864 69202288 8}
HYPERPAR AMETER TUNING TESTS					
Small tests of 5 epochs:					
5/17-18: decay=0.0 005, learningra te=0.015, batchsize =10 on training, 32 on eval, dropout=0 .5 Saved as: "softmax_t est_5_17_1. hdf5"	Train: {'precision': 0.5564379 858060156 } {'recall': 0.4204008 681220477 4} {'total_corr ect': 23499.0} {'acc': 90.109566 30210048, 'f1': 47.894698				

56737692}			
Dev:			
{'precision':			
0.1886792			
452830188			
8}			
{'recall':			
0.0016829			
350387075			
06}			
{'total_corr			
ect':			
5942.0}			
{'acc':			
83.3281414			
275145,			
'f1':			
0.33361134			
27856547}			
Test:			
{'precision':			
0.25}			
{'recall':			
0.0030099			
150141643			
057}			
{'total_corr			
ect':			
5648.0}			
{'acc':			
82.620867			
87983203,			
'f1':			
0.5948215			
535339397			
}			

RUN ON 30: decay=0.0 005, learningrat e=0.015, batchsize= 10 on training, 32 on eval, dropout=0. 5 Saved as: "softmax_t est_5_17_1.h df5"	Seems to be overfitting				
decay=0.05 ,	Overfitting but less? At least on 3 3pochs				
learningrat e=0.0105, batchsize= 10 on training, 32 on eval, dropout=0. 5	Overfitted more				

Trying SGD optimizer with momentum : epochs=3, lr=0.001, lr_decay=lr/nepochs, momentum =0.8, batch_size =10	Overfitting but only slightly-not sure if it ran long enough to test				
Trying SGD optimizer with momentum : epochs=3, lr=0.0105, lr_decay=0.05, momentum =0.8, batch_size =10	Did poorly				
Back to Adam, lr=0.105, decay=0.05 , batch_size =10	F1 of 0, but loss not going up so could be promising over more epochs (compare to two below and see what to implement overnight with)				

5/22 AM 1: Adam, lr=0.015, decay=0.0 05, batch_size =10 (added dropout in other layer), epochs=10	Loss decreasing, acc increasing, val loss up and down (overall up), val acc up and down (overall down)- clearly overfitting still though may be val data size as seems to be doing worse since I integrated it (compare to F1s of: train=49, val=0, test=21				
5/21 overnight: epochs=30 ,	Did poorly				

5/22 AM 3: Adam, lr=0.015, decay=0.0 05, batch_size =10 (added dropout in other layer), epochs=4 + add dropout after activation/ softmax function	Running- worse				
5/22 AM 2: Adam, lr=0.015, decay=0.0 05, batch_size =10 (added dropout in other layer), epochs=4 , same as #1 but use validation split instead of val data and see what happens	Running- success in first two/ three epochs: continued to improve: train=42, dev=1.4, test=1.2				

5/22 PM 4: Adam, lr=0.0105, decay=0.0 005, batch_size =10 (added dropout in other layer), epochs=4 , same as #1 but use validation split instead of val data and see what happens	Running- loss on both was going down and accuracy going up h/ e F1 went from 51 to 1.09 to 1.3				
5/22 PM 6: Adam, lr=0.0105, decay=0.0 005, batch_size =10 , epochs=30 , same as #4 but remove dropout added to word embedding output	<u>Somewhat</u> <u>promising:</u> Performed well on train but less than 1 on dev andf1 of 1 on test so overfitting	Saved on: "softmax_t est_5_22_3. hdf5"			

5/22 PM 5: Adam, lr=0.0105, decay=0.0 005 , epochs=10 , same as #4 but remove dropout added to word embeddin g output— > accidental ly did batch size=4	Running... 48, 20, 17... promising and may want to run longer	Saved on: "softmax_t est_5_22_2. hdf5"			
5/23 1: Adam, lr=0.00105, decay=0.0 005, batch_size =10 , epochs=4 (w/ val split)	<u>Promising</u> : loss going down, accuracy going up on both; <u>Not</u> <u>promising</u> : 45, 0, 0	"softmax_t est_5_23_1. hdf5"			
5/23 2: Adam, lr=0.105, decay=0.0 005, batch_size =10 , epochs=4 (w/ val split)	Could be promising if desperate: no changes. All stayed the same so perhaps not learning? F1s of 0 across.	"softmax_t est_5_23_2. hdf5"			

<u>5/23 3:</u> <u>Adam,</u> <u>lr=0.0105,</u> <u>decay=0.0</u> <u>005,</u> <u>batch_size</u> <u>=10 ,</u> <u>epochs=4</u> <u>(w/ val</u> <u>split)</u>	43, 2, 3.. loss decreasing, acc increasing, could be promising	"softmax_t est_5_23_3. hdf5"			
<u>5/23 4:</u> <u>Adam,</u> <u>lr=0.0105,</u> <u>decay=0.0</u> <u>05,</u> <u>batch_size</u> <u>=10 ,</u> <u>epochs=4</u> <u>(w/ val</u> <u>split)</u>	loss decreasing, acc increasing, could be promising... 40, 0.9, 0.8	"softmax_t est_5_23_4. hdf5"			
<u>5/23 5:</u> <u>Adam,</u> <u>lr=0.0105,</u> <u>decay=0.0</u> <u>0005,</u> <u>batch_size</u> <u>=10 ,</u> <u>epochs=4</u> <u>(w/ val</u> <u>split).</u>	Loss going down, acc slowly going up... 37, 2.6, 2.8. May be worth running for longer	"softmax_t est_5_23_5. hdf5"			
<u>5/23 6:</u> <u>Adam,</u> <u>lr=0.0105,</u> <u>decay=0.0</u> <u>005,</u> <u>batch_size</u> <u>=32 ,</u> <u>epochs=10</u> <u>(w/ val</u> <u>split)</u>	Loss going down, acc going up... 43, 1.18, 1.8	"softmax_t est_5_23_6. hdf5"			

5/23 7: Adam, lr=0.0105, decay=0.0 005 , epochss=2 0, batch size=4 (w/ val split)	Loss going down, acc going up *took 8 hours to train on 20 epochs, 57, 23, 20	"softmax_t est_5_23_7. hdf5"			
5/23 (overnight) 8: Adam, lr=0.0105, decay=0.0 005 , epochs=40 , batch size=4 (w/ val split)	Takes about 15hrs just fyi. 64, 1, 1... overfitting	"softmax_t est_5_23_8. hdf5"			
5/24 1: Adam, lr=0.0105, decay=0.0 005 , epochs=30 , batch size=6 (w/ val split)	Overfit: 65, <1, 1	"softmax_t est_5_24_1. hdf5"			

5/29: 5 epochs, lr=0.0105, lr_decay=0. 0005, batch_size =10, val_split=0. 3	<u>Acc up,</u> <u>loss down,</u> <u>51, 5, 5...</u> <u>some</u> <u>promise...</u> <u>may want</u> <u>to run on</u> <u>more</u> <u>epochs,</u> <u>but clearly</u> <u>overfitting</u> <u>still (play</u> <u>with l2</u> <u>regularizat</u> <u>ion and if</u> <u>no</u> <u>success</u> <u>after</u> <u>playing</u> <u>with</u> <u>different</u> <u>lambda</u> <u>values,</u> <u>play with</u> <u>l1 and /or</u> <u>longer</u> <u>epochs</u> <u>with more</u> <u>dropout</u> <u>layers)</u>	51, 5, 5			
5/29: 5 epochs, lr=0.0105, lr_decay=0. 0005, batch_size =10, val_split=0. 3, l2 added to dense layer where l2=0.01	Acc up, decay down,	39, <1, <1			

5/29: 5 epochs, lr=0.0105, lr_decay=0.0005, batch_size =10, val_split=0.3, l2 added to dense layer where l2=0.001	Loss goes down, accuracy goes up, <u>worth trying over dozens of epochs</u>	46, 2, 1			
5/29: 5 epochs, lr=0.0105, lr_decay=0.0005, batch_size =10, val_split=0.3, l2 added to dense layer where l2=0.0001	Loss goes down, accuracy goes up,	50, <0, <0			

5/29: <u>5 epochs</u> , lr=0.0105, lr_decay=0.0005, batch_size =10, val_split=0.3, l1 added to dense layer where l1=0.01	Performed very poorly, but even performance across test sets so some hope... worth checking out on more epochs, but may need to work on vanishing gradients. Loss goes down for acc and mostly for val, but acc goes down for both.	2.3, 2.3, 2.0			
5/29: <u>3 epochs</u> , lr=0.0105, lr_decay=0.0005, batch_size =10, val_split=0.3, l1 added to dense layer where l1=0.0001	Loss going down on both, acc going up... very promising after only 3 epochs	31.7, 7.26, 10.07			

5/29: Try formatting differently:	<pre>scores = Dense((n_t ags)(x), kernel_regularizer=regularizers.l2 (0.01)) instead of scores = Dense(n_ta gs, activity_regularizer=regularizers.l 1(0.01))(x)</pre>	Doesn't work			
5/29: 3 epochs, lr=0.0105, lr_decay=0. 0005, batch_size =10, val_split=0. 3, l1 added to dense layer where l1=0.001	loss went down, but acc also went down on both... worth looking at over longer epochs	5, 5, 5			

5/29: 5 epochs, $lr=0.0105$, $lr_decay=0.$ 0005 , $batch_size$ $=10$, $val_split=0.$ 3 , l2 added to dense layers where $l2=0.0001$ which were added to the end of both branches.	—not sure this works				
5/29: try with both l1 and l2	Up and down and f1s of 0				
Add ReLU in dense layers (pick best above w/ regularizati on)					
REALIZED ERROR IN PADDING					

5/29: 3 epochs, lr=0.0105, lr_decay=0.0005, batch_size =10, val_split=0.3, l1 added to dense layers where l1=0.001	Loss going down, acc mostly going up but slowly	0, 0, 0			
5/29: 3 epochs, lr=0.0105, lr_decay=0.0005, batch_size =10, val_split=0.3, l1 REMOVED	Loss going down, acc going up	44, 44, 45			
<u>Description of experiment</u>	<u>Summary</u>	<u>Saved as:</u>	<u>Training Results:</u>	<u>Validation and Test Results:</u>	

5/29: 50 epochs, lr=0.0105, lr_decay=0.0005, batch_size =10, val_split=0.3, l1 REMOVED	73, 66, 64... YES. Try to run on 80 epochs and see what happens	"softmax_t est_5_29_1.hdf5"	<pre>{'precision': 0.74971195 60400602} {'recall': 0.7199455 295970041 } {'total_corr ect': 23499.0} {'acc': 95.399295 75043831, 'f1': 73.452729 838272} {'precision': 0.6493506 49350649 3} {'recall': 0.6285410 764872521 } {'total_corr ect': 5648.0} {'acc': 93.166792 29029826, 'f1': 63.877642 82501125}</pre>	<pre>{'precision': 0.6753770 60680463} {'recall': 0.6480982 83406260 5} {'total_corr ect': 5942.0} {'acc': 93.824228 02850357, 'f1': 66.145654 41429062} {'precision': 0.6493506 49350649 3} {'recall': 0.6285410 764872521 } {'total_corr ect': 5648.0} {'acc': 93.166792 29029826, 'f1': 63.877642 82501125}</pre>
---	---	-----------------------------	--	--

5/30: 3 epochs, lr=0.0105, lr_decay=0.0005, batch_size =10, validation_data=dev, l1 REMOVED	Very promising as well and slightly improved performance as compared to val_split. Acc going up, loss going down. Epoch 47 best for loss, 49 best for acc	45,45,47			
--	---	----------	--	--	--

5/30: 50 epochs, lr=0.0105, lr_decay=0.0005, batch_size =10, validation_data=dev, l1 REMOVED	loss: 0.0286 - acc: 0.9906 - val_loss: 0.0319 - val_acc: 0.9903 Loss mostly going down and acc going up though greatly slowed down at end and stayed the same for a few epochs.	"softmax_t est_5_30_1.hdf5"	{'precision': 0.7767666 } {'recall': 0.75918124 } {'total_corr ect': 17549683} {'acc': 96.1708271 } 7401448, 'f1': 76.787328 36913011}	{'precision': 0.69607167 } {'recall': 0.6799057 } 556378324 } {'total_corr ect': 5942.0} {'acc': 94.4122113 } 6248589, 'f1': 68.789375 1064192} {'precision': 0.6530684 } 63994324 3} {'recall': 0.65191218 13031161} {'total_corr ect': 5648.0} {'acc': 93.4532141 } 7034565, 'f1': 65.248981 03845473}	
---	---	-----------------------------	--	--	--

<p>5/30: OVERNIGHT</p> <p>T: 75 epochs, lr=0.0105, lr_decay=0 .0005, batch_size =10, validation_ data=dev, l1 REMOVED</p>	<p>Get test moved to new server</p>	<p>"softmax_t est_5_31_1. hdf5"</p>	<pre>{'precision': 0.8365480 90738886 3} {'recall': 0.8176092 59968509 3} {'total_corr ect': 23499.0} {'acc': 97.1992083 3312871, 'f1': 82.697025 78229243 }</pre>	<pre>{'precision': 0.7537740 760020822 } {'recall': 0.7310669 808145406 } {'total_corr ect': 5942.0} {'acc': 95.317549 9396441, 'f1': 74.224690 3032892} }</pre>	<pre>{'precision': 0.6987951 807228916 } {'recall': 0.6880311 614730878 } {'total_corr ect': 5648.0} {'acc': 94.168192 09647895, 'f1': 69.337139 79837631}</pre>
---	-------------------------------------	---	--	--	--

6/1: OVERNIGH T: 150 epochs, lr=0.0105, lr_decay=0. 0005, batch_size =10, validation_ data=dev, l1 REMOVED	WORSE THAN 75: loss: 0.0327 - acc: 0.9894 - val_loss: 0.0390 - val_acc: 0.9882	"softmax_t est_6_1.hdf 5"	{'precision': 0.7461425 585841968 } {'recall': 0.7140729 392740116 } {'total_corr ect': 23499.0} {'acc': 94.131848 95.374740 3263907, 'f1': 67.593472 72.975558 84143691}	{'precision': 0.6940946 976414258 } {'recall': 0.6587007 741501179} {'total_corr ect': 5942.0} {'acc': 44826915, 'f1': 06631553} {'precision': 0.6453861 676756558 } {'recall': 0.6228753 541076487 } {'total_corr ect': 5648.0} {'acc': 93.1021858 5118983, 'f1': 63.393098 4773403}	
--	---	---------------------------------	---	---	--

6/2: OVERNIGHT	Better than 75 on train, same on validation and test- Epoch 115/115 14041/1404 1 [=====] =====] =====] - 629s 45ms/step - loss: 0.0211 - acc: 0.9929 - val_loss: 0.0291 - val_acc: 0.9916	"softmax_t est_6_2.hdf 5"	{'precision': 0.8722239 123821114} 'recall': 0.8540363 419719988 }'total_corr ect': 23499.0} 'acc': 97.793940 70356201, 'f1': 86.303431 66767008 }' }	{'precision': 0.7537184 365271532 }' 'recall': 0.7334230 898687311 }' 'total_corr ect': 5942.0} 'acc': 95.280557 61068495, 'f1': 74.343227 56738315 } 'precision': 0.6962432 915921288 }' 'recall': 0.6890934 844192634 }' 'total_corr ect': 5648.0} 'acc': 94.144503 06880586, 'f1': 69.26499 37711336 6}
---------------------------	--	---------------------------------	---	---

6/3: OVERNIGHT	BEST YET ON TEST!	"softmax_t est_6_3.hdf 5"	{'precision': 0.8522053 133866392 } 'recall': {'recall': 0.8436103 39145069} 663985701 } 'total_corr ect': 'total_corr ect': 23499.0} 95.726412 'acc': 52287684, 'f1': 76.357773	{'precision': 0.77011297 50085587} 'recall': 0.75715247 39145069} 'total_corr ect': 5942.0} 'acc': 95.726412 'f1': 25186693
T: 100 epochs, lr=0.0105, lr_decay=0 .0005, batch_size =10, validation_ data=dev , I1	Not sure if optimal or a little less or a little more epochs.- Epoch 100/100 14041/1404 1 [===== ===== ===== =====] - 616s 44ms/ step - loss: 0.0224 - acc: 0.9926 - val_loss: 0.0260 - val_acc: 0.9922		84.78860 58895233 2}	{'precision': 0.7235902 926481085 } 'recall': 0.7179532 577903682 } 'total_corr ect': 5648.0} 'acc': 94.570905 56692151, 'f1': 72.076075 36437965

6/4:: 85, lr=0.0105, lr_decay=0 .0005, batch_size =10, validation_ data=dev, l1 REMOVED	Hm didn't perform well but I also got a Jupiter notebook error. Epoch 85/85 1 [===== ===== ===== =====] - 629s 45ms/step - loss: 0.0300 - acc: 0.9902 - val_loss: 0.0308 - val_acc: 0.9905 MAY WANT TO RERUN	"softmax_t est_6_4.hdf 5"-resaved on same weights so will have to rerun anyway	{'precision': 0.78303137 42819266} 'recall': 0.7540746 414741053 }'total_corr ect': 23499.0} 'acc': 95.998939 20568114, 'f1': 76.828025 75386417} 0.6451612} 'precision': 0.6908759 124087591 }' 'recall': 0.6703257 790368272 }' 'total_corr ect': 5648.0} 'acc': 93.883923 76440186, 'f1': 68.044572 25017973}	{'precision': 0.7306000 351926799 }' 'recall': 0.6987546 280713565 }' 'total_corr ect': 5942.0} 'acc': 94.737354 46439001, 'f1': 71.432258 06451612} 'precision': 0.6908759 124087591 }' 'recall': 0.6703257 790368272 }' 'total_corr ect': 5648.0} 'acc': 93.883923 76440186, 'f1': 68.044572 25017973}
---	--	---	--	---

6/7: OVERNIGHT T: 100 epochs, lr=0.0105, lr_decay=0. 0005, batch_size =10, val_split=0. 3, l1 REMOVED	Epoch 100/100 9828/9828 [=====] =====] =====] - 448s 46ms/step - loss: 0.0221 - acc: 0.9926 - val_loss: 0.0326 - val_acc: 0.9902	"softmax_t est_6_7.hdf 5" 5} {'recall': 0.7925869 185922805 } {'total_corr ect': 5942.0} {'acc': 94.676998 96.474332 21524302, 'f1': 70.568392 79.061869 89281544}	{'precision': 0.7886602 30352303 5} {'recall': 0.7041400 201952205 } {'total_corr ect': 55924614, 'f1': 64631472} {'precision': 0.6702683 861972812 } {'recall': 0.6809490 08498583 5} {'total_corr ect': 5648.0} {'acc': 93.853774 09281791, 'f1': 67.556648 51572106}
--	---	--	--

6/8: 150 epochs, lr=0.0105, lr_decay=0.0005, batch_size =10, val_split=0.3, l1 REMOVED	Epoch 150/150 9828/9828 [===== ===== =====] - 440s 45ms/step - loss: 0.0221 - acc: 0.9925 - val_loss: 0.0346 - val_acc: 0.9899	"softmax_t est_6_8.hdf5"	{'precision': 0.7947871 416159861 } {'recall': 0.7785863 228222477 } {'total_corr ect': 23499.0} {'acc': 96.417363 63145255, 'f1': 78.660332 33732452} }	{'precision': 0.7061082 02443281} {'recall': 0.68091551 66610569} {'total_corr ect': 5942.0} {'acc': 4906351, 'f1': 05962988} {'precision': 0.6610598 661602459 } {'recall': 0.64713172 80453258} {'total_corr ect': 5648.0} {'acc': 93.509206 41757295, 'f1': 65.4021651 6059765} }	
6/10: 50 epochs, lr=0.0105, lr_decay=0.0005, batch_size =10, validation_data=dev , l2 ADDED	DID HORRIBLY- prob not implemented correctly	"softmax_t est_6_10.hdf5"			

6/11: 50 epochs, lr=0.0105, lr_decay=0.0005, batch_size =10, validation_split=dev , 2nd fw and bw LSTM ADDED	Epoch 50/50 14041/1404 1 [===== ===== =====] - 610s 43ms/ step - loss: 0.0363 - acc: 0.9881 - val_loss: 0.0370 - val_acc: 0.9883	"softmax_t est_6_11.hdf 5"	{'precision': 0.6557356 131172642} {'recall': 0.6560704 71083876} {'total_corr ect': 23499.0} {'acc': 94.419043 2224574, 65.590299 9361838} {'f1': 504809, 61.855322 02245295} {'precision': 0.59411034 23806989} {'recall': 0.5929532 577903682} {'total_correct': 5648.0} {'acc': 92.652094 32540111, 'f1': 59.353123 61541869}	{'precision': 0.62049110 92294666} {'recall': 0.6166273 981824302} }
--	--	----------------------------	---	--

6/9 morning OVER WEEKEND: 200, lr=0.0105, lr_decay=0 .0005, batch_size =10, validation_ data=dev, I1 REMOVED	Epoch 200/200 14041/1404 1 [=====] =====] =====] =====] 600s 43ms/step - loss: 0.0187 - acc: 0.9938 - val_loss: 0.0295 - val_acc: 0.9916	"softmax_t est_6_9.hdf 5" 8} {'recall': 0.8735690 88046299 8} {'total_corr ect': 5942.0} {'acc': 95.257194 03450022, 'f1': 73.9011925 88.116240 63700556 }	{'precision': 0.8888888 88888888 8} {'recall': 0.7300572 197913161} {'total_corr ect': 5942.0} {'acc': 95.257194 03450022, 'f1': 73.9011925 0425894} {'precision': 0.6985214 569058781 } {'recall': 0.6859065 155807366 } {'total_corr ect': 5648.0} {'acc': 94.056207 60202433, 'f1': 69.2156512 4173664}
---	---	--	--

6/11: 50 epochs, lr=0.0105, lr_decay=0.0005, batch_size=10, validation_split=dev BOTH bidirection al LSTM	Epoch 50/50 14041/1404 1 [====== ====== ====== =====] - 629s 45ms/step - loss: 0.0296 - acc: 0.9903 - val_loss: 0.0311 - val_acc: 0.9901	"softmax_t est_6_11_1.h df5"	{'precision': 0.7637891 254190066 } 'recall': 0.74662751 60645134} 'total_corr ect': 23499.0} 'acc': 95.940988 40492876, 'f1': 75.5110824 187648} 'f1': 68.567526 03722042} 'precision': 0.6611599 928173819 } 'recall': 0.65191218 13031161} 'total_corr ect': 5648.0} 'acc': 93.593194 78841391, 'f1': 65.65035 21440670 4}	{'precision': 0.6957726 957726957 } 'recall': 0.6758667 115449344 } 'total_corr ect': 5942.0} 'acc': 94.386900 8216191, 'f1': 8216191, 68.567526 03722042} 'precision': 0.6611599 928173819 } 'recall': 0.65191218 13031161} 'total_corr ect': 5648.0} 'acc': 93.593194 78841391, 'f1': 65.65035 21440670 4}
---	---	------------------------------------	---	---

100 epochs, lr=0.0105, lr_decay=0. 0005, batch_size =10, validation_ split=dev BOTH bidirection al LSTM	Epoch 100/100 14041/1404 1 [===== ===== =====] - loss: 0.0257 - acc: 0.9915 - val_loss: 0.0288 - val_acc: 0.9911	"softmax_t est_6_13.hd f5"	{'precision': 0.7967360 961992699 } 'recall': 0.71726691 3497139} {'total_corr ect': 5942.0} 'acc': 60161208, 'f1': 72.298558 'f1': 10008482} {'precision': 0.6746436 74115784} 'recall': 0.6788243 62606232 2} 'total_corr ect': 5648.0} 'acc': 94.062668 24593517, 'f1': 67.6727561 5567911}	
--	---	----------------------------------	---	--

300 epochs, lr=0.0105, lr_decay=0.0005, batch_size =10, validation_split=dev BOTH fw and bw LSTM w/ <u>early stopping</u>	Epoch 300/300 14041/1404 1 [===== ===== =====] - 626s 45ms/step - loss: 0.0233 - acc: 0.9924 - val_loss: 0.0353 - val_acc: 0.9900	"softmax_t est_6_14.hdf5"	{'precision': 0.7952248 610409297 } {'recall': 0.80365121 92008171} {'total_corr ect': 23499.0} {'acc': 96.804848 22292397, 'f1': 79.941583 59260905} {'precision': 0.6941391 941391941 } {'recall': 0.70161561 76371592} {'total_corr ect': 5942.0} {'acc': 94.764611 96993885, 'f1': 69.785738 19886172} {'precision': 0.6662621 359223301 } {'recall': 0.6804178 47025495 8} {'total_corr ect': 5648.0} {'acc': 93.858081 18875848, 'f1': 67.326559 21513665}	{'precision': 0.6941391 941391941 } {'recall': 0.70161561 76371592} {'total_corr ect': 5942.0} {'acc': 94.764611 96993885, 'f1': 69.785738 19886172} {'precision': 0.6662621 359223301 } {'recall': 0.6804178 47025495 8} {'total_corr ect': 5648.0} {'acc': 93.858081 18875848, 'f1': 67.326559 21513665}	Most promising epoch numbers: 214, 189, 184 Epoch 189/300 14041/1404 [===== ===== =====] - loss: 0.0267 - acc: 0.9913 - val_loss: 0.0345 - val_acc: 0.9896 Epoch 184/300 14041/1404 [===== ===== =====] - loss: 0.0269 - acc: 0.9912 - val_loss: 0.0342 - val_acc: 0.9896
---	---	---------------------------	--	--	---

6/12 overnight: 100 epochs, lr=0.0105, lr_decay=0. 0005, batch_size =10, validation_ split=dev BOTH bidirection al LSTM w/ RMSprop	Epoch 100/100 14041/1404 1 [===== ===== =====] ===== - 593s 42ms/step - loss: 0.0497 - acc: 0.9869 - val_loss: 0.0399 - val_acc: 0.9889	"softmax_t est_6_12_2. hdf5"	{'precision': 0.67701370 32085561} {'recall': 0.6896038 129282097 } {'total_corr ect': 23499.0} {'acc': 94.766256 91849073, 'f1': 68.325076 42036471} {'precision': 0.6145814 900017696 } {'recall': 0.6149079 320113314} {'total_corr ect': 5648.0} {'acc': 92.942823 30138904, 'f1': 61.474466 76697054}	{'precision': 0.6300712 038416957 } {'recall': 0.6403567 82228206} {'total_corr ect': 5942.0} {'acc': 93.722985 8650364, 'f1': 63.517235 62306986} {'precision': 0.6145814 900017696 } {'recall': 0.6149079 320113314} {'total_corr ect': 5648.0} {'acc': 92.942823 30138904, 'f1': 61.474466 76697054}
--	--	------------------------------------	---	--

6/18 105 epochs, lr=0.0105, lr_decay=0. 0005, batch_size =10, validation_ split=dev (first: two layer LSTM, second: bidirection al)	Epoch 105/105 14041/1404 1 [===== ===== =====] - loss: 0.0230 - acc: 0.9924 - val_loss: 0.0272 - val_acc: 0.9917	"softmax_t est_6_18.hd f5"	{'precision': 0.8290782 279243983 } 'recall': 0.8194816 800714924 } 'total_corr ect': 23499.0} 'acc': 97.1412575 3237633, 'f1': 82.425202 24286265} {'precision': 0.7048828 47433375} 'recall': 0.6977691 218130312} 'total_corr ect': 5648.0} 'acc': 94.293097 87875525, 'f1': 70.1307945 5467567}	
---	--	----------------------------------	--	--

<p>Review GG's preprocessing and if there's regularization besides dropout or batch normalization.</p>	<p><i>Has gradient clipping for anything above or equal to 0; learning rate exponential decay; tries Adam, adagrad, sgd and rmsprop and moves forward with the best; early stopping</i></p>				
<p><u>6/12 Short tests on adagrad, sgd, sgd with momentum, and rmsprop</u></p>					

Test 1: SGD with momentum 0.5, 10 epochs	Epoch 10/10 14041/1404 1 [===== ===== ===== =====] - 602s 43ms/step - loss: 0.1175 - acc: 0.9658 - val_loss: 0.1322 - val_acc: 0.9629	N/A	0, 0, 0		
Test 2: SGD with momentum 0.05, 10 epochs	loss: 0.1309 - acc: 0.9630	N/A	0, 0, 0		
Test 3: SGD with momentum 0.005, 10 epochs	Epoch 10/10 14041/1404 1 [===== ===== ===== =====] - 637s 45ms/step - loss: 0.1129 - acc: 0.9691 - val_loss: 0.1237 - val_acc: 0.9664	N/A	0, 0, 0		

Test 4: adagrad, 10 epochs (same lr and decay) keras.optim izers.Adagr ad(lr=0.010 5, epsilon=No ne, decay=0.0 005)	Epoch 10/10 14041/1404 1 [===== ===== ===== =====] 621s 44ms/ step - loss: 0.0619 - acc: 0.9804 - val_loss: 0.0628 - val_acc: 0.9802 DOES NOT SEEM TO BE OVERFITTI NG SO RECOMME ND RUNNING LONGER TO SEE PERFORMA NCE OVER MORE EPOCHS	"softmax_t est_6_12.hd f5"	{'precision': 0.5079002 218785719} {'recall': 0.3214604 87680326 8} {'total_corr ect': 23499.0} {'acc': 88.1313813 4082438, 'f1': 39.372459 084749295 } {'precision': 0.5199374 511336982 } {'recall': 0.3532223 79603399 4} {'total_corr ect': 5648.0} {'acc': 88.267470 6579089, 'f1': 42.066420 66420664}	{'precision': 0.51762730 83379966} {'recall': 0.31134298 216088857 } {'total_corr ect': 5942.0} {'acc': 87.9658113 0018302, 'f1': 38.881883 144178225 } {'precision': 0.5199374 511336982 } {'recall': 0.3532223 79603399 4} {'total_corr ect': 5648.0} {'acc': 88.267470 6579089, 'f1': 42.066420 66420664}
---	---	----------------------------------	---	---

Test 5: rmsprop, 10 epochs (same lr and decay) keras.optim izers.RMSp rop(lr=0.01 05, rho=0.9, epsilon=No ne, decay=0.0 005)	Epoch 10/10 14041/1404 1 [=====] ===== ===== =====] - 632s 45ms/step - loss: 0.0713 - acc: 0.9817 - val_loss: 0.0544 - val_acc: 0.9835	"softmax_t est_6_12_1.h df5"	{'precision': 0.5040654 183896297 } {'recall': 0.4616792 203923571 5} {'total_corr ect': 23499.0} {'acc': 90.400302 52282428, 'f1': 48.1942161 6098797}	{'precision': 0.4998091 60305343 5} {'recall': 0.4407606 86637495 8} {'total_corr ect': 5942.0} {'acc': 90.097737 62703944, 'f1': 46.843140 76193883} {'precision': 0.4988532 110091743} {'recall': 0.4621104 815864023 } {'total_corr ect': 5648.0} {'acc': 89.858942 60794659, 'f1': 47.9779411 7647059}
TO DO WHEN WORK ON AGAIN:				
Add random seed				

Fix glove embeddings and add preprocessing of line.split etc. with embedding s OR do what DL in Keras book does in 171-172, 203-205, 232					
Get interactive shell to work					
Batch normalization					
Play with batch size over long epochs					
Test 6: Adam, 10 epochs (same lr and decay)					
Test out different activation functions in different layers?					
If no success, rerun 85 epoch test above					

Other ideas:					
Add gradient clipping (do not think I need but gg and papers use)					
CBOW?					
Less layers?					
Are there such things as sparse layers in keras? If so try vs dense					
Add ngrams or more dense layers?					
Add POS tag preprocessing or layer					
Subsampling?					
Dynamic context windows?					
Context distribution smoothing?					
Context vectors?					
FastText w/ or instead of Glove?					

learningrate=Triangular cyclical, batchsize=10 on training, 32 on eval, dropout=0.5 —> also not working for whatever reason	Did not work when tried. Need to revisit				
---	--	--	--	--	--

<p>2 epochs, decay=0.0 005, learningrat e=annealin g, batchsize= 10 on training, 32 on eval, dropout=0. 5: there's cosine annealing- also suggest making each cycle longer than the previous one by constant factor T_mul (look up "Sgdr: Stochastic gradient descent with restarts")</p>				
<p>Huang and Yixuan Li [7] inspired by SGDR wrote a follow-up paper "Snapshots Ensemble: Train 1, get M for free</p>				

Starting with small lr and then slowly increasing (not sure how to do this as it is opposite of decay) (look up Leslie N. Smith)					
Use "clip norm or "clipvalue" for gradient clipping (paper uses value of 5.0; article says recommended default is 1.0)- do not need to use while using bi LSTMs	N/A				
K-fold validation					
Add CNN?					
Finish CRF and test					