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

Tests:	**—-Acc(from eval)—-(TRAIN)**	**—F1—(TRAIN)**	**—Classes Predicted— (TRAIN)**	**Loss on last epoch/val loss (TRAIN)**	**Precision/ Recall (TRAIN)**
100 epochs, decay=0.9, batchsize=32, learningrate=0.0001, dropout=0.5	~83	.13/100	NA	NA but continued to go down	
300 epochs, decay=0.9, batchsize=32, learningrate=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 improvement 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	0.04163197335553705 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, learningrate=0.00001, batchsize=32, dropout=0.5:	3.2457	4.464203019357229/100	0, 1, 2, 3, 4, 5, 6, 7, 8	2.2873/2.2802	
2 epochs, decay=0.1, learningrate=0.0001, batchsize=32, dropout=0.5:	~83	0	0, 9	0.3089/0.3149	
2 epochs, decay=0.5, learningrate=0.0001, batchsize=32, dropout=0.5:	~83	0.00831981363617455/100	0, 5, 6, 9	1.7557/1.7766	
NER Paper: 2 epochs, decay=0.05, learningrate=0.015, batchsize=32, dropout=0.5:	85.6	26.394331418755705/100	0, 2, 4, 6, 8, 9	0.0745/0.0745	

NER Paper: 20 epochs, decay=0.05, learningrate=0.015, batchsize=32, dropout=0.5:	88.5	**39.6859786404302 5**	0, 2, 4, 6, 8, 9	0.0568/0.0622	{‘precision': 0.4947883564255751 7} {'recall': 0.3312906932209881 4}
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': 0.2347759479126771 4}**
***NER Paper: 30 epochs, decay=0.05, learningrate=0.015, batchsize=32, dropout=0.5:**	**88.7879933798576 8**	**41.2831453825053 4**	0, 2, 4, 6, 8, 9	0.0535/0.0606	**{'precision': 0.5120015 12001512} {'recall': 0.3458445040214477 3}**

***NER Paper: 40 epochs, decay=0.05, learningrate=0.015, batchsize=32, dropout=0.5:**	**87.3023902249768**	35.91309517375561	0, 2, 4, 6, 8, 9	0.0607/0.0666	**{'precision': 0.4919671281557711} {'recall': 0.282777990552789 46}**
***NER Paper: 2 epochs, decay=0.05, learningrate=0.015, batchsize=10 on training, 32 on eval, dropout=0.5:**	85.96804848222924	**29.60019463141676**	0, 2, 4, 6, 8, 9	0.0707/0.0729	{'precision': 0.40573588261449534} {'recall': 0.2329886378143751 }
NER Paper: 20 epochs, decay=0.05, learningrate=0.015, batchsize=10 on training, 32 on eval, dropout=0.5:	86.91490563350538	34.71209348162748	0, 2, 4, 6, 8, 9	0.0638/0.0697	{'precision': 0.512982689747004} {'recall': 0.262309034426996 9}

***NER Paper: 2 epochs, decay=0.05, learningrate=0.0105, batchsize=10 on training, 32 on eval, dropout=0.5: **<u>h/e precision much better with bachsize=32 so may want to run longer with that and test</u>**	86.4198682847054	**31.63851799451764 5**	Forgot to capture.	0.0694/0.0727	{'precision': 0.514672036823935 6} {'recall': 0.2283926975615983 6}
3)NER Paper: 20 epochs, decay=0.05, learningrate=0.0105, batchsize=10 on training, 32 on eval, dropout=0.5:	87.87895158161486	37.48813165945775	0, 2, 4, 6, 8, 9	0.0592/0.0647	{'precision': 0.492959700353749 03} {'recall': 0.3024384016341120 6}

NER Paper: 2 epochs, decay=0.05, learningrate=0.0105, batchsize=32 on training, 32 on eval, dropout=0.5:	85.70284990251497	27.645291221788888	0, 2, 6, 8, 9	0.0787/0.0848	**{'precision': 0.5555698316844404} {'recall': 0.184007830120430 65}**
****NER Paper: 20 epochs, decay=0.05, learningrate=0.0105, batchsize=32 on training, 32 on eval, dropout=0.5:**	88.6706	**40.875324217471224**	**0, 2, 4, 6, 8, 9**	0.0552/0.0612	{'precision': 0.5006168270416975} {'recall': 0.34537639899570194}
NER Paper: 30 epochs, decay=0.05, learningrate=0.0105, batchsize=32 on training, 32 on eval, dropout=0.5:	87.21006183055775	36.552806143513806	0, 2, 4, 6, 8, 9	0.0620/0.0683	{'precision': 0.5174563591022444} {'recall': 0.2825652155410869}

NER Paper: 2 epochs, decay=0.05, learningrate=0.015, batchsize=32 on training, 32 on eval, dropout=0.68:	84.85470555591024	20.297211554516615	0, 2, 6, 8, 9	0.0806/0.0870	{'precision': 0.4714551659944151} {'recall': 0.12932465211285588}
NER Paper: 2 epochs, decay=0.05, learningrate=0.015, batchsize=10 on training, 32 on eval, dropout=0.68:	85.36791391850545	26.96844315444945	0, 2, 6, 8, 9	0.0817/0.0858	{'precision': 0.4913919207831664} {'recall': 0.18583769522107324}
NER Paper: 2 epochs, decay=0.05, learningrate=0.0105, batchsize=10 on training, 32 on eval, dropout=0.68:	85.52163087304353	27.659041052961452	0, 2, 6, 8, 9	0.0798/0.0852	{'precision': 0.5246700749019142} {'recall': 0.18779522532873738}

~~2 epochs, decay=0.05, learningrate=0.0105, batchsize=20 on training, 32 on eval, dropout=0.5:~~	86.3560241821816	30.972359328726558	0, 2, 4, 6, 8, 9	0.0742/0.0772	{'precision': 0.5629696086127621} {'recall': 0.21362611174943616}
2 epochs, decay=0.005, learningrate=0.0105, batchsize=10 on training, 32 on eval, dropout=0.5:	86.88347469072444	34.89317533085373	0, 2, 4, 6, 8, 9	0.0671/0.0702	{'precision': 0.5410407725321889} {'recall': 0.25750031916251753}
20 epochs, decay=0.005, learningrate=0.0105, batchsize=10 on training, 32 on eval, dropout=0.5:	91.11732090501471	53.01448608875604	0, 2, 4, 6, 8, 9	0.0430/0.0505	{'precision': 0.5766594967735481} {'recall': 0.4905740669815737}

50 epochs, decay=0.005, learningrate=0.0105, batchsize=10 on training, 32 on eval, dropout=0.5:	**92.93196674213367**	**62.10457780471925 6**	0, 2, 4, 6, 8, 9	0.0343/0.0459	{'precision': 0.6573044387415645 } {'recall': 0.588578237371803}
~~100 epochs, decay=0.005, learningrate=0.0105, batchsize=10 on training, 32 on eval, dropout=0.5:~~	92.47228920396226	59.62555962555963	0, 2, 4, 6, 8, 9	0.0357/0.0471	{'precision': 0.6361267911419887} {'recall': 0.561087705859823 8}
20 epochs, decay=0.005, learningrate=0.0105, batchsize=20 on training, 32 on eval, dropout=0.5:					

2 epochs, decay=0.0005, learningrate=0.0105, batchsize=10 on training, 32 on eval, dropout=0.5:	**89.19561341904813**	**43.9222078833571**	0, 2, 4, 6, 8, 9	0.0602/0.0589	{'precision': 0.46927914852443153} {'recall': 0.41278352270309376}
20 epochs, decay=0.0005, learningrate=0.0105, batchsize=10 on training, 32 on eval, dropout=0.5:	**93.9608390097288 6**	**65.52232336367577**	0, 2, 3, 4, 6, 8, 9	0.0346/0.0397	{'precision': 0.6676383551963253 } {'recall': 0.6432614153793779}
~~20 epochs, decay=0.0005, learningrate=0.0105, batchsize=32 on training, 32 on eval, dropout=0.5:~~	92.24490597728132	56.874972933177425	0, 2, 4, 6, 8, 9	0.0399/0.0467	{'precision': 0.5789798527531632 } {'recall': 0.5588748457381165 }

50 epochs, decay=0.0005, learningrate=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.**	92.79642080139082	59.30134223822018	0, 2, 3, 4, 6, 8, 9	0.0357/0.0449	{'precision': 0.6045627376425855} {'recall': 0.5818971020043406}
75 epochs, decay=0.0005, learningrate=0.0105, batchsize=10 on training, 32 on eval, dropout=0.5:	**96.237617927424**	**78.4562064096529 9**	0, 2, 3, 4, 6, 8, 9	**0.0204/0.0368**	{'precision': 0.7888826743535688} {'recall': 0.7802885229158688}

~~2 epochs, decay=0.00005, learningrate=0.0105, batchsize=10 on training, 32 on eval, dropout=0.5:~~	86.4621036140673	32.5443950399072	0, 2, 4, 6, 8, 9	0.0724/0.0736	{'precision': 0.46357199055861525} {'recall': 0.25073407379037405}
2 epochs, decay=0.0005, learningrate=0.01005, batchsize=10 on training, 32 on eval, dropout=0.5:	88.0208819326101	37.24704946293594	0, 2, 4, 6, 8, 9	0.0639/0.0644	{'precision': 0.49429818386597213} {'recall': 0.29882122643516745}
5 epochs, decay=0.0005, learningrate=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**	88.55520795988626	40.16627562990921	0, 2, 4, 6, 8, 9	0.0582/0.0608	{'precision': 0.5011773690574683} {'recall': 0.33512064343163545}

2 epochs, decay=0.0005, learningrate=0.00105, batchsize=10 on training, 32 on eval, dropout=0.5:	87.65009502949106	36.73844739060464	0, 2, 4, 6, 8, 9	0.0632/0.0659	{'precision': 0.45954754601226994} {'recall': 0.3060130218307162}
*2 epochs, decay=0.0005, learningrate=0.010005, batchsize=10 on training, 32 on eval, dropout=0.5:	89.24570648410527	43.91148082009087	0, 2, 4, 6, 8, 9	0.0596/0.0580	{'precision': 0.46832208293153327} {'recall': 0.4133367377335206}
20 epochs, decay=0.0005, learningrate=0.010005, batchsize=10 on training, 32 on eval, dropout=0.5:	92.54890212699082	58.17207112970711	0, 2, 3, 4, 6, 8, 9	0.0385/0.0445	{'precision': 0.5961409620795927} {'recall': 0.5679816162389889}

75 epochs, decay=0.0005, learningrate=0.0105, batchsize=10 on training, 32 on eval, dropout=0.5 w/ dev(validation data) in training: 5/17/18	Train: 96.629 dev: 95.34	Train: 79.728 dev: 74.032			Train: {'precision': 0.8077830188679245} {'recall': 0.7870547682880122} Dev: {'precision': 0.7550306211723534} {'recall': 0.726186469202288 8}
HYPERPARAMETER TUNING TESTS					
Small tests of 5 epochs:					
5/17-18: decay=0.0005, learningrate=0.015, batchsize=10 on training, 32 on eval, dropout=0.5 Saved as: "softmax_test_5_17_1.hdf5"	Train: {'precision': 0.5564379858060156} {'recall': 0.4204008681220477 4} {'total_correct': 23499.0} {'acc': 90.10956630210048, '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': 0.3336113427856547} 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=0.0005, learningrate=0.015, batchsize=10 on training, 32 on eval, dropout=0.5 Saved as: "softmax_test_5_17_1.hdf5"	Seems to be overfitting				
decay=0.05, learningrate=0.0105, batchsize=10 on training, 32 on eval, dropout=0.5	Overfitting but less? At least on 3 3pochs				
decay=0.0005, learningrate=0.001, 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=0.005, 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, lr=0.00105, lr_decay=0.005, batch_size=10	Did poorly				

5/22 AM 3: Adam, lr=0.015, decay=0.005, 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=0.005, 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- success in first two/ three epochs: continued to imrove: train=42, dev=1.4, test=1.2				

5/22 PM 4: Adam, lr=0.0105, decay=0.0005, 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=0.0005, batch_size=10 , epochs=30 , same as #4 but remove dropout added to word embedding output	Somewhat promising: Performed well on train but less than 1 on dev andf1 of 1 on test so overfitting	Saved on: "softmax_test_5_22_3.hdf5"			

5/22 PM 5: Adam, lr=0.0105, decay=0.0005 , epochs=10 , same as #4 but remove dropout added to word embedding output—> accidentally did batch size=4	Running… 48, 20, 17… promising and may want to run longer	Saved on: "softmax_test_5_22_2.hdf5"			
5/23 1: Adam, lr=0.00105, decay=0.0005, batch_size=10 , epochs=4 (w/ val split)	Promising: loss going down, accuracy going up on both; Not promising: 45, 0, 0	"softmax_test_5_23_1.hdf5"			
5/23 2: Adam, lr=0.105, decay=0.0005, 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_test_5_23_2.hdf5"			

5/23 3: Adam, lr=0.0105, decay=0.0005, batch_size=10 , epochs=4 (w/ val split)	43, 2, 3.. loss decreasing, acc increasing, could be promising	"softmax_test_5_23_3.hdf5"			
5/23 4: Adam, lr=0.0105, decay=0.005, batch_size=10 , epochs=4 (w/ val split)	loss decreasing, acc increasing, could be promising... 40, 0.9, 0.8	"softmax_test_5_23_4.hdf5"			
5/23 5: Adam, lr=0.0105, decay=0.00005, batch_size=10 , epochs=4 (w/ val split)	Loss going down, acc slowly going up... 37, 2.6, 2.8. May be worth running for longer	"softmax_test_5_23_5.hdf5"			
5/23 6: Adam, lr=0.0105, decay=0.005, batch_size=32 , epochs=10 (w/ val split)	Loss going down, acc going up... 43, 1.18, 1.8	"softmax_test_5_23_6.hdf5"			

5/23 7: Adam, lr=0.0105, decay=0.0005 , epochs=20, batch size=4 (w/ val split)	Loss going down, acc going up *took 8 hours to train on 20 epochs, 57, 23, 20	"softmax_test_5_23_7.hdf5"			
5/23 (overnight) 8: Adam, lr=0.0105, decay=0.0005 , epochs=40, batch size=4 (w/ val split)	Takes about 15hrs just fyi. 64, 1, 1... overfitting	"softmax_test_5_23_8.hdf5"			
5/24 1: Adam, lr=0.0105, decay=0.0005 , epochs=30, batch size=6 (w/ val split)	Overfit: 65, <1, 1	"softmax_test_5_24_1.hdf5"			

5/29: 5 epochs, lr=0.0105, lr_decay=0.0005, batch_size=10, val_split=0.3	**Acc up, loss down, 51, 5, 5... some promise... may want to run on more epochs, but clearly overfitting still (play with l2 regularization and if no success after playing with different lambda values, play with l1 and /or longer epochs with more dropout layers)**	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, **worth trying over dozens of epochs**	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: 5 epochs, 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: 3 epochs, 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:~~	~~*scores = Dense((n_tags)(x), kernel_regularizer=regularizers.l2(0.01)) instead of scores = Dense(n_tags, activity_regularizer=regularizers.l1(0.01))(x)*~~	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/ regularization)					
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			
Description of experiment	**Summary**	**Saved as:**	**Training Results:**	**Validation and Test Results:**	

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_test_5_29_1.hdf5"	{'precision': 0.7497119560400602} {'recall': 0.7199455295970041} {'total_correct': 23499.0} {'acc': 95.39929575043831, 'f1': 73.452729838272}	{'precision': 0.675377060680463} {'recall': 0.648098283406260 5} {'total_correct': 5942.0} {'acc': 93.82422802850357, 'f1': 66.14565441429062} {'precision': 0.649350649350649 3} {'recall': 0.628541076487252 1} {'total_correct': 5648.0} {'acc': 93.16679229029826, 'f1': 63.87764282501125}	

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_test_5_30_1.hdf5"	{'precision': 0.7767666652153089} {'recall': 0.7591812417549683} {'total_correct': 23499.0} {'acc': 96.17082717401448, 'f1': 76.78732836913011}	{'precision': 0.6960716747070985} {'recall': 0.6799057556378324} {'total_correct': 5942.0} {'acc': 94.41221136248589, 'f1': 68.78937510641923} {'precision': 0.653068463994324 3} {'recall': 0.6519121813031161} {'total_correct': 5648.0} {'acc': 93.45321417034565, 'f1': 65.24898103845473}	

5/30: OVERNIGHT: 75 epochs, lr=0.0105, lr_decay=0.0005, batch_size=10, validation_data=dev, l1 REMOVED	Get test moved to new server	"softmax_test_5_31_1.hdf5"	{'precision': 0.836548090738886 3} {'recall': 0.817609259968509 3} {'total_correct': 23499.0} {'acc': 97.199208333312871, **'f1': 82.69702578229243** }	{'precision': 0.753774076002082 2} {'recall': 0.731066980814540 6} {'total_correct': 5942.0} {'acc': 95.3175499396441, **'f1': 74.2246903032892**} {'precision': 0.698795180722891 6} {'recall': 0.688031161473087 8} {'total_correct': 5648.0} {'acc': 94.16819209647895, **'f1': 69.33713979837631**}	

6/1: OVERNIGHT: 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_test_6_1.hdf5"	{'precision': 0.7461425585841968} {'recall': 0.7140729392740116} {'total_correct': 23499.0} {'acc': 95.3747403263907, 'f1': 72.97555884143691}	{'precision': 0.6940946976414258} {'recall': 0.6587007741501179} {'total_correct': 5942.0} {'acc': 94.13184844826915, 'f1': 67.59347206631553} {'precision': 0.6453861676756558} {'recall': 0.6228753541076487} {'total_correct': 5648.0} {'acc': 93.10218585118983, 'f1': 63.3930984773403}	

6/2: OVERNIGHT: 115 epochs, lr=0.0105, lr_decay=0.0005, batch_size=10, validation_data=dev , l1 REMOVED	Better than 75 on train, same on validation and test- Epoch 115/115 14041/14041 [==============================] - 629s 45ms/step - loss: 0.0211 - acc: 0.9929 - val_loss: 0.0291 - val_acc: 0.9916	"softmax_test_6_2.hdf5"	{'precision': 0.8722239123821114} {'recall': 0.8540363419719988} {'total_correct': 23499.0} {'acc': 97.79394070356201, '**f1': 86.30343166767008**}	{'precision': 0.7537184365271532} {'recall': 0.7334230898687311} {'total_correct': 5942.0} {'acc': 95.28055761068495, **'f1': 74.34322756738315**} {'precision': 0.6962432915921288} {'recall': 0.6890934844192634} {'total_correct': 5648.0} {'acc': 94.14450306880586, **'f1': 69.2649937711336 6**}	

6/3: OVERNIGHT: 100 epochs, lr=0.0105, lr_decay=0.0005, batch_size=10, validation_data=dev , l1 REMOVED	BEST YET ON TEST! Not sure if optimal or a little less or a little more epochs.- Epoch 100/100 14041/14041 [==============================] - 616s 44ms/step - loss: 0.0224 - acc: 0.9926 - val_loss: 0.0260 - val_acc: 0.9922	"softmax_test_6_3.hdf5"	{'precision': 0.8522053133866392} {'recall': 0.8436103663985701} {'total_correct': 23499.0} {'acc': 97.48159570967 63, **'f1': 84.7886058895233 2}**	{'precision': 0.7701129750085587} {'recall': 0.7571524739145069} {'total_correct': 5942.0} {'acc': 95.72641252287684, **'f1': 76.35777325186693** } {'precision': 0.7235902926481085 } {'recall': 0.7179532577903682 } {'total_correct': 5648.0} {'acc': 94.57090556692151, **'f1': 72.07607536437965** }	

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 14041/14041 [==============================] - 629s 45ms/step - loss: 0.0300 - acc: 0.9902 - val_loss: 0.0308 - val_acc: 0.9905 **MAY WANT TO RERUN**	"softmax_test_6_4.hdf5"-resaved on same weights so will have to rerun anyway	{'precision': 0.7830313742819266} {'recall': 0.7540746414741053} {'total_correct': 23499.0} {'acc': 95.99893920568114, 'f1': 76.82802575386417}	{'precision': 0.7306000351926799} {'recall': 0.6987546280713565} {'total_correct': 5942.0} {'acc': 94.73735446439001, 'f1': 71.43225806451612} {'precision': 0.6908759124087591} {'recall': 0.6703257790368272} {'total_correct': 5648.0} {'acc': 93.88392376440186, 'f1': 68.04457225017973}	

6/7: OVERNIGHT: 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_test_6_7.hdf5"	{'precision': 0.788660230352303 5} {'recall': 0.7925869185922805 } {'total_correct': 23499.0} {'acc': 96.47433221524302, 'f1': 79.06186989281544}	{'precision': 0.7072346179851251} {'recall': 0.7041400201952205 } {'total_correct': 5942.0} {'acc': 94.67699855924614, 'f1': 70.56839264631472} {'precision': 0.6702683861972812 } {'recall': 0.6809490084985835} {'total_correct': 5648.0} {'acc': 93.85377409281791, 'f1': 67.55664851572106}	

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_test_6_8.hdf5"	{'precision': 0.7947871416159861} {'recall': 0.7785863228222477} {'total_correct': 23499.0} {'acc': 96.41736363145255, 'f1': 78.66033233732452}	{'precision': 0.706108202443281} {'recall': 0.6809155166610569} {'total_correct': 5942.0} {'acc': 94.5154004906351, 'f1': 69.32830705962988} {'precision': 0.6610598661602459} {'recall': 0.6471317280453258} {'total_correct': 5648.0} {'acc': 93.50920641757295, 'f1': 65.40216516059765}	
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_test_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 14041/14041 [==========] - 610s 43ms/step - loss: 0.0363 - acc: 0.9881 - val_loss: 0.0370 - val_acc: 0.9883	"softmax_test_6_11.hdf5"	{'precision': 0.6557356131172642} {'recall': 0.656070471083876} {'total_correct': 23499.0} {'acc': 94.4190432224574, 'f1': 65.5902999361838}	{'precision': 0.6204911092294666} {'recall': 0.6166273981824302} {'total_correct': 5942.0} {'acc': 93.35695650480 9, 'f1': 61.85532202245295} {'precision': 0.5941103423806989} {'recall': 0.5929532577903682} {'total_correct': 5648.0} {'acc': 92.65209432540111, 'f1': 59.35312361541869}	

6/9 morning OVER WEEKEND: 200, lr=0.0105, lr_decay=0.0005, batch_size=10, validation_data=dev, l1 REMOVED	Epoch 200/200 14041/14041 [==========] - 600s 43ms/step - loss: 0.0187 - acc: 0.9938 - val_loss: 0.0295 - val_acc: 0.9916	"softmax_test_6_9.hdf5"	{'precision': 0.888888888888888} {'recall': 0.873569088046299} {'total_correct': 23499.0} {'acc': 98.08418581580486, '**f1': 88.11624063700556**}	{'precision': 0.7481890307002415} {'recall': 0.7300572197913161} {'total_correct': 5942.0} {'acc': 95.25719403450022, 'f1': 73.90119250425894} {'precision': 0.6985214569058781} {'recall': 0.6859065155807366} {'total_correct': 5648.0} {'acc': 94.05620760202433, 'f1': 69.21565124173664}	

6/11: **50 epochs, lr=0.0105, lr_decay=0.0005, batch_size=10, validation_split=dev BOTH bidirectional LSTM**	Epoch 50/50 14041/14041 [==========] - 629s 45ms/step - loss: 0.0296 - acc: 0.9903 - val_loss: 0.0311 - val_acc: 0.9901	"softmax_test_6_11_1.hdf5"	{'precision': 0.7637891254190066} {'recall': 0.7466275160645134} {'total_correct': 23499.0} {'acc': 95.94098840492876, 'f1': 75.5110824187648}	{'precision': 0.6957726957726957} {'recall': 0.6758667115449344} {'total_correct': 5942.0} {'acc': 94.38690082161 91, 'f1': 68.56752603722042} {'precision': 0.6611599928173819} {'recall': 0.6519121813031161} {'total_correct': 5648.0} **{'acc': 93.59319478841391, 'f1': 65.6503521440670 4}**	

100 epochs, lr=0.0105, lr_decay=0.0005, batch_size=10, validation_split=dev BOTH bidirectional LSTM	Epoch 100/100 14041/14041 [==============================] - 622s 44ms/step - loss: 0.0257 - acc: 0.9915 - val_loss: 0.0288 - val_acc: 0.9911	"softmax_test_6_13.hdf5"	{'precision': 0.7967360961992699} {'recall': 0.7894804034214222} {'total_correct': 23499.0} {'acc': 96.72234199812397, 'f1': 79.30916552667578}	{'precision': 0.728796169630643} {'recall': 0.717266913497139} {'total_correct': 5942.0} {'acc': 95.06055060161208, 'f1': 72.29855810008482} {'precision': 0.674643674115784} {'recall': 0.678824362606232 2} {'total_correct': 5648.0} {'acc': 94.06266824593517, 'f1': 67.67275615567911}	

300 epochs, lr=0.0105, lr_decay=0.0005, batch_size=10, validation_split=dev BOTH fw and bw LSTM w/ early stopping	Epoch 300/300 14041/14041 [==========] - 626s 45ms/step - loss: 0.0233 - acc: 0.9924 - val_loss: 0.0353 - val_acc: 0.9900	"softmax_test_6_14.hdf5"	{'precision': 0.7952248610409297} {'recall': 0.8036512192008171} {'total_correct': 23499.0} {'acc': 96.80484822292397, 'f1': 79.94158359260905}	{'precision': 0.6941391941391941} {'recall': 0.7016156176371592} {'total_correct': 5942.0} {'acc': 94.76461196993885, 'f1': 69.78573819886172} {'precision': 0.6662621359223301} {'recall': 0.6804178470254958} {'total_correct': 5648.0} {'acc': 93.85808118875848, 'f1': 67.32655921513665}	Most promising epoch numbers: 214, 189, 184 **Epoch 189/300** 14041/14041 [==========] - 625s 44ms/step - loss: 0.0267 - acc: 0.9913 - val_loss: 0.0345 - val_acc: 0.9896 **Epoch 184/300** 14041/14041 [==========] - 625s 45ms/step - 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 bidirectional LSTM w/ RMSprop	Epoch 100/100 14041/14041 [==============================] - 593s 42ms/step - loss: 0.0497 - acc: 0.9869 - val_loss: 0.0399 - val_acc: 0.9889	"softmax_test_6_12_2.hdf5"	{'precision': 0.6770137032085561} {'recall': 0.6896038129282097} {'total_correct': 23499.0} {'acc': 94.76625691849073, 'f1': 68.32507642036471}	{'precision': 0.6300712038416957} {'recall': 0.640356782228206} {'total_correct': 5942.0} {'acc': 93.7229858650364, 'f1': 63.51723562306986} {'precision': 0.6145814900017696} {'recall': 0.6149079320113314} {'total_correct': 5648.0} {'acc': 92.94282330138904, 'f1': 61.47446676697054}	

Review GG's preprocessing and if there's regularization besides dropout or batch normalization.	*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*				
6/12 Short tests on adagrad, sgd, sgd with momentum, and rmsprop					

Test 1: SGD with momentum 0.5, 10 epochs	Epoch 10/10 14041/14041 [==============================] - 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 14041/14041 [==============================] - 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.optimizers.Adagrad(lr=0.0105, epsilon=None, decay=0.0005)	Epoch 10/10 14041/14041 [==============================] - 621s 44ms/step - loss: 0.0619 - acc: 0.9804 - val_loss: 0.0628 - val_acc: 0.9802 DOES NOT SEEM TO BE OVERFITTING SO RECOMMEND RUNNING LONGER TO SEE PERFORMANCE OVER MORE EPOCHS	"softmax_test_6_12.hdf5"	{'precision': 0.5079002218785719} {'recall': 0.321460487680326 8} {'total_correct': 23499.0} {'acc': 88.13138134082438, 'f1': 39.372459084749295 }	{'precision': 0.5176273083379966} {'recall': 0.3113429821608885 7} {'total_correct': 5942.0} {'acc': 87.965811300183 02, 'f1': 38.881883144178225 } {'precision': 0.5199374511336982 } {'recall': 0.3532223796033994} {'total_correct': 5648.0} {'acc': 88.2674706579089, 'f1': 42.06642066420664}	

Test 5: rmsprop, 10 epochs (same lr and decay) keras.optimizers.RMSprop(lr=0.0105, rho=0.9, epsilon=None, decay=0.0005)	Epoch 10/10 14041/14041 [==============================] - 632s 45ms/step - loss: 0.0713 - acc: 0.9817 - val_loss: 0.0544 - val_acc: 0.9835	"softmax_test_6_12_1.hdf5"	{'precision': 0.5040654183896297} {'recall': 0.46167922039235715} {'total_correct': 23499.0} {'acc': 90.40030252282428, 'f1': 48.19421616098797}	{'precision': 0.499809160305343 5} {'recall': 0.440760686637495 8} {'total_correct': 5942.0} {'acc': 90.09773762703944, 'f1': 46.84314076193883} {'precision': 0.4988532110091743} {'recall': 0.4621104815864023} {'total_correct': 5648.0} {'acc': 89.85894260794659, 'f1': 47.97794117647059}	
TO DO WHEN WORK ON AGAIN:					
Add random seed					

Fix glove embeddings and add preprocessing of line.split etc. with embeddings 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				

2 epochs, decay=0.0005, learningrate=annealing, 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")	""				
Huang and Yixuan Li [7] inspired by SGDR wrote a follow-up paper "Snapshots Ensemble: Train 1, get M for free					

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					