Best configuration:

Configuration	Loss Value	Accuracy

1 hidden layer, with cube non linearity and with below best configuration :
 	
`max_iter = 10001`
`batch_size = 2000`
`hidden_size = 800`
`embedding_size = 50`
`learning_rate = 0.2`
`display_step = 100`
`validation_step = 200`
`n_Tokens = 48`
`lam = 1e-8`
	`Average loss at step 9900 : 0.1332571244984865`
	`Average loss at step 10000 : 0.13224200427532196`
		`Testing on dev set at step 10000`
		`UAS: 86.0109180647`
		`UASnoPunc: 87.8228678008`
		`LAS: 83.6553082234`
		`LASnoPunc: 85.1579720794`
		
		`UEM: 29.8235294118`
		`UEMnoPunc: 32.3529411765`
		`ROOT: 85.8235294118`

Below is the output for best configuration:

```
Average loss at step  0 :  4.55938196182251
Average loss at step  100 :  3.1485586130619048
Average loss at step  200 :  0.8381325000524521
Average loss at step  300 :  0.5521899256110191
Average loss at step  400 :  0.44473945409059523
Average loss at step  500 :  0.3878600573539734
Average loss at step  600 :  0.3468229228258133
Average loss at step  700 :  0.3339368750154972
Average loss at step  800 :  0.3006752997636795
Average loss at step  900 :  0.2868726049363613
Average loss at step  1000 :  0.2848645755648613
Average loss at step  1100 :  0.26870464146137235
Average loss at step  1200 :  0.25744102761149407
Average loss at step  1300 :  0.25393659859895706
Average loss at step  1400 :  0.2518626822531223
Average loss at step  1500 :  0.2303618736565113
Average loss at step  1600 :  0.2420640005171299
Average loss at step  1700 :  0.22381655678153037
Average loss at step  1800 :  0.22500980138778687
Average loss at step  1900 :  0.222765311896801
Average loss at step  2000 :  0.21758420661091804
Average loss at step  2100 :  0.218398324996233
Average loss at step  2200 :  0.20951731517910957
Average loss at step  2300 :  0.20717422053217888
Average loss at step  2400 :  0.20897179827094078
Average loss at step  2500 :  0.20035785242915152
Average loss at step  2600 :  0.2064592131972313
Average loss at step  2700 :  0.198370743393898
Average loss at step  2800 :  0.1908312928676605
```

```
Average loss at step  2900 :  0.19971832886338234
Average loss at step  3000 :  0.18912159502506257
Average loss at step  3100 :  0.19008327066898345
Average loss at step  3200 :  0.1878118548542261
Average loss at step  3300 :  0.19054663635790348
Average loss at step  3400 :  0.17887178175151347
Average loss at step  3500 :  0.1905478949844837
Average loss at step  3600 :  0.1786206253618002
Average loss at step  3700 :  0.17942456543445587
Average loss at step  3800 :  0.1819731480628252
Average loss at step  3900 :  0.17568452201783658
Average loss at step  4000 :  0.1773345421999693
Average loss at step  4100 :  0.17405981749296187
Average loss at step  4200 :  0.17254422813653947
Average loss at step  4300 :  0.17530052028596402
Average loss at step  4400 :  0.17002836383879186
Average loss at step  4500 :  0.17680487677454948
Average loss at step  4600 :  0.16876387365162374
Average loss at step  4700 :  0.16221200965344906
Average loss at step  4800 :  0.17301519677042962
Average loss at step  4900 :  0.16367990992963313
Average loss at step  5000 :  0.16391607627272606
Average loss at step  5100 :  0.16213813230395316
Average loss at step  5200 :  0.16725621730089188
Average loss at step  5300 :  0.15600236751139163
Average loss at step  5400 :  0.1690804872661829
Average loss at step  5500 :  0.15708980187773705
Average loss at step  5600 :  0.15794896744191647
Average loss at step  5700 :  0.16128380946815013
Average loss at step  5800 :  0.15599589429795743
Average loss at step  5900 :  0.15808545671403407
Average loss at step  6000 :  0.1534740746021271
Average loss at step  6100 :  0.1553532962501049
Average loss at step  6200 :  0.15628626741468907
Average loss at step  6300 :  0.15292647570371629
Average loss at step  6400 :  0.159779899045825
Average loss at step  6500 :  0.1504961483925581
Average loss at step  6600 :  0.14605122201144696
Average loss at step  6700 :  0.15614281445741654
Average loss at step  6800 :  0.15025863699615002
Average loss at step  6900 :  0.148619227707386
Average loss at step  7000 :  0.14713837653398515
Average loss at step  7100 :  0.152147404178977
Average loss at step  7200 :  0.14231618233025073
Average loss at step  7300 :  0.15311114974319934
Average loss at step  7400 :  0.14311823569238186
Average loss at step  7500 :  0.1430524070560932
Average loss at step  7600 :  0.14808522626757623
Average loss at step  7700 :  0.14518261432647706
Average loss at step  7800 :  0.14465571135282518
Average loss at step  7900 :  0.14267587281763552
Average loss at step  8000 :  0.14273056402802467
Average loss at step  8100 :  0.14146880477666854
Average loss at step  8200 :  0.14166488990187645
```

```
Average loss at step  8300 :  0.14651176132261753
Average loss at step  8400 :  0.1389019648730755
Average loss at step  8500 :  0.13553422197699547
Average loss at step  8600 :  0.14464459590613843
Average loss at step  8700 :  0.13911730416119097
Average loss at step  8800 :  0.13984663866460323
Average loss at step  8900 :  0.13581565111875535
Average loss at step  9000 :  0.14068999327719212
Average loss at step  9100 :  0.13236116215586663
Average loss at step  9200 :  0.1423726489394903
Average loss at step  9300 :  0.13314303264021873
Average loss at step  9400 :  0.13278742313385009
Average loss at step  9500 :  0.13896012112498282
Average loss at step  9600 :  0.13652212999761104
Average loss at step  9700 :  0.13333721928298473
Average loss at step  9800 :  0.13459827832877636
Average loss at step  9900 :  0.1332571244984865
Average loss at step  10000 :  0.13224200427532196

Testing on dev set at step  10000
UAS: 86.0109180647
UASnoPunc: 87.8228678008
LAS: 83.6553082234
LASnoPunc: 85.1579720794

UEM: 29.8235294118
UEMnoPunc: 32.3529411765
ROOT: 85.8235294118

Train Finished.
```

Experiments:

1. Number of hidden layers

Configuration	Loss Value	Accuracy
1 hidden layer, with cube non linearity, 5000 iterations	Average loss at step 4900 : 0.20033312901854516 Average loss at step 5000 : 0.20044935420155524	Testing on dev set at step 5000 UAS: 81.5564473914 UASnoPunc: 83.5980331204 LAS: 78.4929082434 LASnoPunc: 80.1559938959 UEM: 22.4705882353 UEMnoPunc: 23.8823529412 ROOT: 81.7058823529

2 hidden layers, both with cube non linearity, 5000 iterations	Average loss at step 4900 : 0.19238572597503661 Average loss at step 5000 : 0.19166974782943724	Testing on dev set at step 5000 UAS: 14.7219383304 UASnoPunc: 15.0539761488 LAS: 1.16908043971 LASnoPunc: 1.3027751088 UEM: 0.941176470588 UEMnoPunc: 0.941176470588 ROOT: 7.17647058824
2 hidden layers, first with cube and second with tanh non linearity, 5000 iterations	Average loss at step 4900 : 0.19790953680872916 Average loss at step 5000 : 0.19636754125356673	Testing on dev set at step 5000 UAS: 8.50262980781 UASnoPunc: 8.81139433674 LAS: 0.366428197522 LASnoPunc: 0.412592550726 UEM: 1.05882352941 UEMnoPunc: 1.11764705882 ROOT: 6.17647058824
3 hidden layers, with cube non linearity, 5000 iterations	Average loss at step 4900 : 2.3568258142471312 Average loss at step 5000 : 2.4024523186683653	Testing on dev set at step 5000 UAS: 16.3646334472 UASnoPunc: 16.1758887696 LAS: 2.90151307426 LASnoPunc: 3.28943650031 UEM: 0.588235294118 UEMnoPunc: 0.588235294118 ROOT: 2.58823529412

3 hidden layers, first with cube second with relu and third with tanhnon linearity, 5000 iterations	Average loss at step 4900 : 0.19220773205161096 Average loss at step 5000 : 0.19140200778841973	Testing on dev set at step 5000 UAS: 22.4966971608 UASnoPunc: 24.5718645792 LAS: 0.565844903657 LASnoPunc: 0.641496637088 UEM: 0.941176470588 UEMnoPunc: 0.941176470588 ROOT: 3.05882352941

It is clear from the above table that the accuracy is going down as the number of layers increases thus it is preferred to use one hidden layer.

2.

(a) Try sigmoid, tanh, and ReLU.

Configuration	Loss Value	Accuracy
with cube non linearity, 5000 iterations	Average loss at step 4900 : 0.20033312901854516 Average loss at step 5000 : 0.20044935420155524	Testing on dev set at step 5000 UAS: 81.5564473914 UASnoPunc: 83.5980331204 LAS: 78.4929082434 LASnoPunc: 80.1559938959 UEM: 22.4705882353 UEMnoPunc: 23.8823529412 ROOT: 81.7058823529

with sigmoid non linearity, 5000 iterations	Average loss at step 4900 : 0.44703450322151184 Average loss at step 5000 : 0.4382682463526726	Testing on dev set at step 5000 UAS: 69.0305855373 UASnoPunc: 72.0482676765 LAS: 63.6538125982 LASnoPunc: 66.1787147459 UEM: 9.05882352941 UEMnoPunc: 9.47058823529 ROOT: 60.8235294118
with tanh non linearity, 5000 iterations	Average loss at step 4900 : 0.2608226631581783 Average loss at step 5000 : 0.2590936607122421	Testing on dev set at step 5000 UAS: 78.2561009049 UASnoPunc: 80.4951110609 LAS: 74.8036991799 LASnoPunc: 76.6093935455 UEM: 17.6470588235 UEMnoPunc: 18.8235294118 ROOT: 76.2941176471
with relu non linearity, 5000 iterations	Average loss at step 4900 : 0.2538101543486118 Average loss at step 5000 : 0.2506736005842686	Testing on dev set at step 5000 UAS: 77.7351247601 UASnoPunc: 80.0033911716 LAS: 74.4547199442 LASnoPunc: 76.3324478607 UEM: 17.0 UEMnoPunc: 17.8823529412 ROOT: 77.1176470588

It is clear from the above table that the accuracy is best with cube non linearity.

(c) Effect of fixing Word, POS and Dep Embeddings – The paper allowed POS

and Dep embeddings to be learnt and the Word embeddings to be modified via backprop. You can fix these (use a bit vector to uniquely index each POS and Dep category) and use the pre-trained word embeddings without change, and see how your performance varies.

Testing is done on cube non linearity

Configuration	Loss Value	Accuracy
without fixing Word, POS and Dep Embeddings	Average loss at step 4900 : 0.20033312901854516 Average loss at step 5000 : 0.20044935420155524	Testing on dev set at step 5000 UAS: 81.5564473914 UASnoPunc: 83.5980331204 LAS: 78.4929082434 LASnoPunc: 80.1559938959 UEM: 22.4705882353 UEMnoPunc: 23.8823529412 ROOT: 81.7058823529
with fixing Word, POS and Dep Embeddings	Average loss at step 4900 : 0.7833374255895614 Average loss at step 5000 : 0.788374879360199	Testing on dev set at step 5000 UAS: 9.2205299499 UASnoPunc: 9.71570677669 LAS: 0.51848343595 LASnoPunc: 0.587803085966 UEM: 1.11764705882 UEMnoPunc: 1.23529411765 ROOT: 6.52941176471

It is clear from the above table that the accuracy is going down when fixing word, pos and dep embeddings.

(d) Best Configuration – Use the dev set to find the best model.

Tuning is done on the hyper parameters in config.py to get the best configuration **(with cube non linearity)**

Configuration	Loss Value	Accuracy
max_iter = 1001 batch_size = 10000 hidden_size = 200 embedding_size = 50 learning_rate = 0.1 display_step = 100 validation_step = 200 n_Tokens = 48 lam = 1e-8	Average loss at step 900 : 0.37036251068115233 Average loss at step 1000 : 0.35361800909042357	Testing on dev set at step 1000 UAS: 66.9317247052 UASnoPunc: 70.3696377098 LAS: 62.6218311439 LASnoPunc: 65.9780704233 UEM: 7.05882352941 UEMnoPunc: 7.64705882353 ROOT: 47.1764705882
with max iterations = 5001 max_iter = 5001 batch_size = 10000 hidden_size = 200 embedding_size = 50 learning_rate = 0.1 display_step = 100 validation_step = 200 n_Tokens = 48 lam = 1e-8	Average loss at step 4900 : 0.20033312901854516 Average loss at step 5000 : 0.20044935420155524	Testing on dev set at step 5000 UAS: 81.5564473914 UASnoPunc: 83.5980331204 LAS: 78.4929082434 LASnoPunc: 80.1559938959 UEM: 22.4705882353 UEMnoPunc: 23.8823529412 ROOT: 81.7058823529
with max iterations = 10001 max_iter = 10001 batch_size = 10000 hidden_size = 200 embedding_size = 50 learning_rate = 0.1 display_step = 100 validation_step = 200 n_Tokens = 48 lam = 1e-8	Average loss at step 9900 : 0.16402560383081435 Average loss at step 10000 : 0.16319277197122573	Testing on dev set at step 10000 UAS: 84.4230625421 UASnoPunc: 86.158367716 LAS: 81.8406161976 LASnoPunc: 83.24761205 UEM: 27.8235294118 UEMnoPunc: 29.4117647059 ROOT: 86.4705882353

with batch size = 40000 max_iter = 1001 batch_size = 40000 hidden_size = 200 embedding_size = 50 learning_rate = 0.1 display_step = 100 validation_step = 200 n_Tokens = 48 lam = 1e-8	Average loss at step 900 : 0.3700397038459778 Average loss at step 1000 : 0.35101267367601396	Testing on dev set at step 1000 UAS: 70.5386743774 UASnoPunc: 73.5432091788 LAS: 66.6276142284 LASnoPunc: 69.292940711 UEM: 9.70588235294 UEMnoPunc: 10.4117647059 ROOT: 59.0588235294
with batch size = 2000 max_iter = 1001 batch_size = 2000 hidden_size = 200 embedding_size = 50 learning_rate = 0.1 display_step = 100 validation_step = 200 n_Tokens = 48 lam = 1e-8	Average loss at step 900 : 0.3605086126923561 Average loss at step 1000 : 0.3532983139157295	Testing on dev set at step 1000 UAS: 71.652915223 UASnoPunc: 74.5633866501 LAS: 67.5449310766 LASnoPunc: 70.0729101905 UEM: 10.7058823529 UEMnoPunc: 11.5882352941 ROOT: 63.0
with batch size = 1000 max_iter = 1001 batch_size = 1000 hidden_size = 200 embedding_size = 50 learning_rate = 0.1 display_step = 100 validation_step = 200 n_Tokens = 48 lam = 1e-8	Average loss at step 900 : 0.3962704066932201 Average loss at step 1000 : 0.34820873245596884	Testing on dev set at step 1000 UAS: 71.0521723957 UASnoPunc: 73.9784095405 LAS: 67.0638382731 LASnoPunc: 69.6433617815 UEM: 10.7647058824 UEMnoPunc: 11.4705882353 ROOT: 61.9411764706

with hidden size = 100
 	
`max_iter = 1001`
`batch_size = 10000`
`hidden_size = 100`
`embedding_size = 50`
`learning_rate = 0.1`
`display_step = 100`
`validation_step = 200`
`n_Tokens = 48`
`lam = 1e-8`
	Average loss at step 900 : 0.3866528618335724
	Average loss at step 1000 : 0.3624024108052254
		Testing on dev set at step 1000
		UAS: 66.8370017698
		UASnoPunc: 70.3300740406
		LAS: 62.4697759055
		LASnoPunc: 65.6502571639
		
		UEM: 7.88235294118
		UEMnoPunc: 8.41176470588
		ROOT: 47.5882352941

with hidden size = 400
 	
`max_iter = 1001`
`batch_size = 10000`
`hidden_size = 400`
`embedding_size = 50`
`learning_rate = 0.1`
`display_step = 100`
`validation_step = 200`
`n_Tokens = 48`
`lam = 1e-8`
	Average loss at step 900 : 0.3591331523656845
	Average loss at step 1000 : 0.33597402155399325
		Testing on dev set at step 1000
		UAS: 67.4227883441
		UASnoPunc: 70.7228847567
		LAS: 63.4369469302
		LASnoPunc: 66.6308709659
		
		UEM: 8.41176470588
		UEMnoPunc: 8.94117647059
		ROOT: 48.2352941176

with hidden size = 800
 	
`max_iter = 1001`
`batch_size = 10000`
`hidden_size = 800`
`embedding_size = 50`
`learning_rate = 0.1`
`display_step = 100`
`validation_step = 200`
`n_Tokens = 48`
`lam = 1e-8`
	`Average loss at step 900 : 0.3491692292690277`
	`Average loss at step 1000 : 0.3308147317171097`
		`Testing on dev set at step 1000`
		`UAS: 71.2366328489`
		`UASnoPunc: 74.2270954615`
		`LAS: 67.2632549792`
		`LASnoPunc: 69.9400893009`
		
		`UEM: 10.7058823529`
		`UEMnoPunc: 11.3529411765`
		`ROOT: 58.4117647059`

with learning rate = 0.2
 	
`max_iter = 1001`
`batch_size = 10000`
`hidden_size = 200`
`embedding_size = 50`
`learning_rate = 0.2`
`display_step = 100`
`validation_step = 200`
`n_Tokens = 48`
`lam = 1e-8`
	Average loss at step 900 : 0.30475007444620134
	Average loss at step 1000 : 0.2890088349580765
		Testing on dev set at step 1000
		UAS: 73.4102749458
		UASnoPunc: 76.3211439552
		LAS: 69.9179898796
		LASnoPunc: 72.4919459673
		
		UEM: 12.0
		UEMnoPunc: 12.8823529412
		ROOT: 60.8823529412

with learning rate = 0.05
 	
`max_iter = 1001`
`batch_size = 10000`
`hidden_size = 200`
`embedding_size = 50`
`learning_rate = 0.05`
`display_step = 100`
`validation_step = 200`
`n_Tokens = 48`
`lam = 1e-8`
	Average loss at step 900 : 0.47925644397735595
	Average loss at step 1000 : 0.45303824633359907
		Testing on dev set at step 1000
		UAS: 63.4219906773
		UASnoPunc: 66.865427005
		LAS: 58.2421417354
		LASnoPunc: 61.3604250268
		
		UEM: 5.76470588235
		UEMnoPunc: 6.29411764706
		ROOT: 42.9411764706

Best configuration for hyper parameters found from above table is -

```
max_iter = 10001
batch_size = 2000
hidden_size = 800
embedding_size = 50
learning_rate = 0.2
display_step = 100
validation_step = 200
n_Tokens = 48
lam = 1e-8
```

1) As number of iterations increases, loss decreases and accuracy increases.
2) As batch size decreases, loss decreases and accuracy increases (but if batch size is below 2000 then it results in over-fitting)
3) As hidden size increases, loss decreases and accuracy increases.
4) As learning rate increases, loss decreases and accuracy increases.

(e) gradient clipping

What is gradient clipping and why is it required?
Propagation happens in both forward and backward modes in multi layered neural networks.
In case of backward propagation, if gradients get multiplied with numbers **>1 (greater than 1)** , then there is a possibility for gradients to explode i.e they get exponentially so large creating **exploding gradients problem** . Also, if gradients get multiplied with numbers **<1 (less than 1)** , then there is a possibility for gradients to vanish (**called as vanishing gradient problem**)

gradient clipping rescues by clipping gradients between 2 numbers, preventing them from getting either too small or too large.

Configuration	Loss Value	Accuracy
With gradient clipping	Average loss at step 4900 : 0.20033312901854516 Average loss at step 5000 : 0.20044935420155524	Testing on dev set at step 5000 UAS: 81.5564473914 UASnoPunc: 83.5980331204 LAS: 78.4929082434 LASnoPunc: 80.1559938959 UEM: 22.4705882353 UEMnoPunc: 23.8823529412 ROOT: 81.7058823529
Without gradient clipping	Average loss at step 0 : 4.4932756423950195 Average loss at step 100 : nan Average loss at step 200 : nan	**Testing failed**

We can see in the above table that without gradient clipping the values of loss resulted into NAN and we couldn't check the accuracy.

Findings and observations:

1) As number of hidden layers increases, accuracy goes down. Average loss for 1 and 2 hidden layers are very near, but loss is much lower for 1 hidden layer when compared with 3 hidden layers (cubic)

2) Cube has lower loss and higher accuracy when compared to sigmoid, tanh and

relu.

3) As number of iterations increases, loss decreases and accuracy increases. As batch size decreases, loss decreases and accuracy increases (but if batch size is below 2000 then it results in over-fitting). As hidden size increases, loss decreases and accuracy increases. As learning rate increases, loss decreases and accuracy increases.

4) Gradient clipping rescues by clipping gradients between 2 numbers, preventing them from getting either too small or too large