Report on relation extraction

Yiqing Hua

December 9, 2015

1 Oct. 20th

1.1 General Settings

The following results are on trained on the same hyperparameter settings. Currently they are using the logistic regression without any regularization. Some settings include:

- OCLASS weight:

	Target	Agent	DSE
weight	0.3	0.8	0.5

- Network Layers: 2
- Learning Rate of the Classifiers: 0.1
- Word Vector Dimension: 25
- Tested and Trained on Bishan's data and her datasplits.

1.2 Experiments

$$prediction = sigmoid(\theta_{arg}V_{arg} + \theta_{dse}V_{dse} + b)$$

Variant 1 The training samples are picked according to the labels predicted. While training, the spans that have no overlap with any gold standard answers are not included. No backpropagation to the neural networks. The feature vectors are extracted from the last hidden layer of the neural network. Each span has two feature vectors, the average of the forward hidden layer vectors and the average of the backward hidden layer vectors.
The results on entity extractions are shown as follows.
And the Relation results:

	Target		Agent		DSE	
	Prop.	Bin.	Prop.	Bin.	Prop.	Bin.
P	0.278156	0.317073	0.575763	0.588529	0.347986	0.411924
R	0.46463	0.587744	0.512373	0.536866	0.542221	0.561512
F1	0.347986	0.411924	0.590153	0.65977	0.537502	0.576856

	P	R	F1
is from	0.300813	0.318052	0.309192
is about	0.289109	0.418338	0.34192

Variant 2 The training samples are picked according to the labels predicted. While training, the spans that have no overlap with any gold standard answers are not included. **There's backpropagation to the neural networks.**
The feature vectors are extracted from the last hidden layer of the neural network. Each span has one feature vector, the concatenation of the forward hidden layer vectors from the first token and the backward hidden layer vector from last token.
The results on entity extractions are shown as follows.
And the Relation results:

	Target		Agent		DSE	
	Prop.	Bin.	Prop.	Bin.	Prop.	Bin.
P	0.283911	0.320681	0.591958	0.607143	0.524426	0.540486
R	0.478922	0.612813	0.518698	0.543779	0.556322	0.616092
F1	0.35649	0.421036	0.552912	0.573717	0.539903	0.575818

	P	R	F1
is from	0.312227	0.409742	0.354399
is about	0.312	0.446991	0.367491

Variant 3 The training samples are picked according to the **gold standard labels**. There's backpropagation to the neural networks. **And the learning rate on the output layer of the neural network does not decay**.
The feature vectors are extracted from the last hidden layer of the neural network. Each span has one feature vector, the concatenation of the forward hidden layer vectors from the first token and the backward hidden layer vector from last token.
The results on entity extractions are shown as follows.

	Target		Agent		DSE	
	Prop.	Bin.	Prop.	Bin.	Prop.	Bin.
P	0.266633	0.303738	0.606677	0.621083	0.514708	0.533719
R	0.490078	0.62117	0.47435	0.497696	0.577337	0.641379
F1	0.345365	0.407982	0.532415	0.552585	0.544227	0.582617

And the Relation results:

	P	R	F1
is from	0.302752	0.378223	0.336306
is about	0.3	0.378223	0.334601

1.3 Some Conclusion

- According to my other experiments, the relation classifier simply predicts most of the span pairs as true pairs. So it may be possible that it's backpropagating useless errors to the neural net. But this also indicates that the recall we have right now is the **upper bound** of recall we can ever get.

- Since the classifier can not tell the neural net some spans are false spans, maybe it will not improve the entity extraction results as we expected.

- Because now we have more information to backpropagate to the last hidden layer, tuning the decay rate may also be effective but sometimes it can also lead to gradient explosion.

I'm also trying to use the library Bishan used for linear regression, which is liblinear. And I tried to tune the weight of the classifier so it will not only make trivial suggestions.
However, the training results of the classifier doesn't improve with more epochs with neural network. The accuracy is from 40% to 60%. I'm training on gold standard pairs, among which around 58% of the agent dse pairs are related, and around 55% of the target dse pairs are related.

2 Oct. 27th

2.1 Experiments

- Tried a bilinear model for classification.
- Tune the weights of the classifier error.

2.2 Bilinear Model

$$prediction = sigmoid(V_{arg}^T \theta V_{dse} + b)$$

	Target		Agent		DSE	
	Prop.	Bin.	Prop.	Bin.	Prop.	Bin.
P	0.288455	0.337739	0.586474	0.6	0.512167	0.537002
R	0.469011	0.571031	0.543636	0.569124	0.599119	0.650575
F1	0.357214	0.42444	0.564243	0.584155	0.552241	0.588358

Variant 1 Slight improvement (1 to 2%) on target and expression, agent extraction doesn't improve.

And the Relation results: Still have very high recall and predicts almost

	P	R	F1
is from	0.320088	0.415473	0.361596
is about	0.218359	0.464497	0.297067

every pair as true pair.

Variant 2 +adjusted the weights of the classifier to prevent learning trivial results.

	Target		Agent		DSE	
	Prop.	Bin.	Prop.	Bin.	Prop.	Bin.
P	0.286985	0.32967	0.570199	0.581776	0.498337	0.51751
R	0.456446	0.579387	0.54209	0.569124	0.556379	0.609195
F1	0.352402	0.42023	0.555789	0.575381	0.525761	0.559622

The extraction results get a little bit worse. And the Relation results get even more worse:

	P	R	F1
is from	0.286842	0.312321	0.29904
is about	0.163136	0.227811	0.190123

Variant 3 +lower the entity extraction error rate to 0.8.

	Target		Agent		DSE	
	Prop.	Bin.	Prop.	Bin.	Prop.	Bin.
P	0.231616	0.267081	0.579255	0.594458	0.484868	0.505415
R	0.467996	0.626741	0.509163	0.541475	0.596111	0.641379
F1	0.309873	0.37455	0.541952	0.566731	0.534765	0.565337

Recall of the target has been increased. And the Relation results don't have many changes.

	P	R	F1
is from	0.321311	0.280802	0.299694
is about	0.156364	0.127219	0.140294

Variant 4 +L1 regularization to classifier.
Gradient of the target extraction exploded.

2.3 Explosive Gradient

When training with liblinear(L1-regularized logistic regression, to ensure no bugs in logistic regression). By observing the prediction accuracy on training

set of the classifier, it goes like (44%, 55%, 57%, 44%, 55%, ...) on agent-dse relation and the similar pattern for target-dse relation. Whenever the accuracy drops, it's the sign for that one of the neural network has an explosive gradient problem. Usually, the network cannot recover from the explosive gradient, but the program actually re-initialize the weights if the explosion occurs.(explosion would output the weight as -nan(Not a Number in C++), but when reloading, the network would not load something as nan(hence the re-initialize))
The upper bound of the accuracy rate ever reached by the classifier is similar to the percentage of true span pairs in the training set.(But by looking at the output results from the classifier, it's not predicting everything as true.)
Gradient clipping would produce worse results.
The explosive gradient happens before in :

- Multitasking RNN.
- Using trained dse RNN to initialize target extraction RNN.

L1-regularized logistic regression + gradient clipping

	Target		Agent		DSE	
	Prop.	Bin.	Prop.	Bin.	Prop.	Bin.
P	0.308669	0.380561	0.639876	0.642984	0.575967	0.588792
R	0.413031	0.536528	0.355715	0.421388	0.423227	0.491968
F1	0.353304	0.445282	0.457243	0.509119	0.487922	0.536043

Recall of the target has been increased. And the Relation results don't have many changes.

	P	R	F1
is from	0.357401	0.283668	0.316294
is about	0.223881	0.0443787	0.0740741

3 Nov. 6th

I implemented Adagrad. It may prevent the gradient explosion, however the neural network tends to predict everything as a true label. The training accuracy on relation classification has been improved, can reach 60% for IS-FROM relation sometimes. The final results get worse, even if I tune the OWEIGHT all to 1.
Here are the results.

	Target		Agent		DSE	
	Prop.	Bin.	Prop.	Bin.	Prop.	Bin.
P	0.0894239	0.396432	0.0599972	0.0599972	0.0443143	0.448651
R	1	1	1	1	1	1
F1	0.164167	0.567778	0.113203	0.113203	0.0848678	0.619405

Recall of the target has been increased. And the Relation results don't have many changes.

	P	R	F1
is from	0.0312094	0.0687679	0.0429338
is about	0.143885	0.0591716	0.0838574

Sometimes the gradients still explode but not all the time. I've also tried to lower the weights of relation error, the pattern doesn't change much.

No backprop from relation – doesn't work.
Tried adding adagrad switch -¿ but still no results, should be the same as drnt.cpp in pipeline. (see diff.out)
no backprop in diff.out: should have no explosion, however explodes.(see current results folder in pipeline.)

Conclusion after debuging:
Trained with no relation: Preventing explosive gradients needs a very carefully tuned initialization point. But adagrad can prevent the gradients from explosion. However, the result is worse than a good initialization.
random parameter of initialization: 135
Results with Adagrad:

	Target		Agent		DSE	
	Prop.	Bin.	Prop.	Bin.	Prop.	Bin.
P	0.313894	0.360805	0.60151	0.605495	0.520701	0.537563
R	0.370669	0.489395	0.324086	0.376771	0.371509	0.425703
F1	0.339927	0.415376	0.421222	0.464503	0.433631	0.475138

Results without Adagrad:

Target extraction suffers explosion.

Improvement in relation identification improves the results in entity extraction slightly.

	Target		Agent		DSE	
	Prop.	Bin.	Prop.	Bin.	Prop.	Bin.
P			0.52048	0.527163	0.506992	0.526077
R			0.538566	0.592166	0.476092	0.528736
F1			0.529369	0.557777	0.491056	0.527403

From folder record/results4 and record/results5. The parameter tuning of the relation classifier improves the relation extraction results a little bit, and so do the entity extraction.

results4:

	Target		Agent		DSE	
	Prop.	Bin.	Prop.	Bin.	Prop.	Bin.
P	0.288143	0.322888	0.444579	0.458763	0.403597	0.425041
R	0.404176	0.548747	0.549625	0.585253	0.513716	0.570115
F1	0.336436	0.406555	0.491553	0.514346	0.452047	0.487004

	P	R	F1
is from	0.666667	0.00573066	0.0113636
is about	1	0	0

	Pairs predicted as true	Pairs selected out of all
is from	3/909	145/349
is about	0/1151	88/338

results5:

	Target		Agent		DSE	
	Prop.	Bin.	Prop.	Bin.	Prop.	Bin.
P	0.283443	0.315722	0.453226	0.466783	0.405829	0.424483
R	0.41631	0.562674	0.545015	0.582949	0.524138	0.581609
F1	0.337262	0.404484	0.4949	0.518439	0.457458	0.490777

	P	R	F1
is from	0.267913	0.246418	0.256716
is about	0.111111	0.00591716	0.011236

	Pairs predicted as true	Pairs selected out of all
is from	321/904	145/349
is about	15/1256	89/338

results6:
relation classification training on candidates:

	target		agent		dse	
	prop.	bin.	prop.	bin.	prop.	bin.
p	0.307961	0.337719	0.457281	0.470483	0.411651	0.429936
r	0.396059	0.534819	0.545007	0.585253	0.533793	0.595402
f1	0.346498	0.414007	0.497305	0.52163	0.464832	0.499318

	p	r	f1
is from	1	0	0
is about	0.1	0.00295858	0.00574713

	pairs predicted as true	pairs selected out of all
is from	0/890	148/349
is about	9/1189	86/338

However in this setting, backprop seems to worse the extraction results on dse. Though in most cases the precision gets slightly higher because of backprop. (The relation classification result also gets worse, though it maybe the problem caused by the classification parameter). results8(same setting, no backprop):

	target		agent		dse	
	prop.	bin.	prop.	bin.	prop.	bin.
p	0.293498	0.331006	0.439916	0.452736	0.41524	0.434572
r	0.415197	0.54039	0.56108	0.599078	0.538774	0.597701
f1	0.343898	0.410542	0.493164	0.515727	0.469009	0.503247

	p	r	f1
is from	0.16129	0.415473	0.232372
is about	0.0678138	0.198225	0.101056

	pairs predicted as true	pairs selected out of all
is from	899/948	153/349
is about	988/1178	94/338

Looks like the relation classifier is only performing trivially.

TEST ON RELATION hasn't been updated every 5 epoch. seems like that backprop changes a lot in the relation classification space.

experiment sets:

- Training set of relation classifier: A. all picked out candidates, B. gold standard, C. A + B.
- Regarding candidates: A. all of them, B. only those having overlapping with gold standard.

Objective of relation classifier:

- Should not act trivially on precision and recall
- Ideally picking the right pairs out of the wrong ones.

4 Nov. 19th

I'm currently playing with the liblinear settings, trying to see if the variants bring any differences. I fixed the initialization point by using the random seed, and I'm using adagrad in every experiment.

4.1 Experiment 1

experiment on training set of relation classifier:
A. all candidates
B. 10% of the no overlapping candidates
I retrain the relation classifier every 5 epochs, but don't update them during the training of the neural networks. In group A, the results of the relation classifier don't change a lot. The results of relation classification I'm showing is from epoch 190 to epoch 195.
But the relation classification results differ a lot after 5 epochs of training of neural network without tuning in groupB. Say from epoch 190 to epoch 195.

GroupA:

Table 1: Entity Extraction Results

	target		agent		dse	
	prop.	bin.	prop.	bin.	prop.	bin.
p	0.302324	0.327422	0.415808	0.429467	0.43753	0.45922
r	0.397719	0.543175	0.567095	0.610599	0.505479	0.570115
f1	0.343522	0.408564	0.479809	0.504261	0.469056	0.508694

Table 2: Relation classification in epoch 195

	p	r	f1
is from	0.231263	0.309456	0.264706
is about	0.15873	0.118343	0.135593

Table 3: Relation classification in epoch 190

	p	r	f1
is from	0.22547	0.309456	0.26087
is about	0.164062	0.12426	0.141414

GroupB:

Table 4: Results of Entity Extraction

	target		agent		dse	
	prop.	bin.	prop.	bin.	prop.	bin.
p	0.305391	0.334734	0.412603	0.426128	0.431529	0.451376
r	0.397797	0.537604	0.562257	0.608295	0.48636	0.547126
f1	0.345523	0.412579	0.475943	0.501171	0.457307	0.49466

Table 5: Results of Relation Classification in Epoch 195

	p	r	f1
is from	0	0	0
is about	1	0	0

Table 6: Results of Relation Classification in Epoch 190

	p	r	f1
is from	0.232184	0.289398	0.257653
is about	0.212291	0.112426	0.147002

4.2 Experiment 2

continue from expr1, 10% of the no overlapping candidates
Raise the REL_WEIGHT
A. from expr1/groupB
B. REL_WEIGHT = 0.8
Which means that more error from the classifier are backpropagated to the hidden layers.
Results from GroupB:

Table 7: Results of Entity Extraction

	target		agent		dse	
	prop.	bin.	prop.	bin.	prop.	bin.
p	0.298395	0.329167	0.41473	0.4288	0.433548	0.454044
r	0.401019	0.532033	0.555781	0.596774	0.491418	0.556322
f1	0.342178	0.406706	0.475005	0.499031	0.460672	0.500006

Table 8: Results of Relation Classification in Epoch 195

	p	r	f1
is from	0.6	0.00859599	0.0169492
is about	1	0	0

Table 9: Results of Relation Classification in Epoch 190

	p	r	f1
is from	0.231343	0.266476	0.24767
is about	0.179641	0.0887574	0.118812

Some example of the backpropgate errors.(the squared norm of the error matrixes from a randomly picked sentence from the same epoch.)
GroupA:
Target gpyd:0.671458
Agent gpyd:0.290214
DSE gpyd:0.29489
Agent backprop err:0.0810047 0.10472
DSE backprop err:0.135083 0.0887444
Target backprop err:0.0110059 0.0161923
DSE backprop err:0.00746071 0.00419274
GourpB:
Target gpyd:0.665647
Agent gpyd:0.308587
DSE gpyd:0.298005
Agent backprop err:0.219774 0.0930358
DSE backprop err:0.357541 0.355972
Target backprop err:0.0145334 0.0272726
DSE backprop err:0.0119703 0.0163344

4.3 Experiment 3

continue from expr1
setting same as groupA
Raise the REL_WEIGHT
A. from expr1/GroupA
B. REL_WEIGHT = 1

Results from GroupB:

Table 10: Results of Entity Extraction

	target		agent		dse	
	prop.	bin.	prop.	bin.	prop.	bin.
p	0.301173	0.324866	0.412207	0.425697	0.434729	0.455497
r	0.392476	0.545961	0.56327	0.610599	0.50931	0.574713
f1	0.340815	0.407347	0.476042	0.501652	0.469074	0.508207

Table 11: Results of Relation Classification in Epoch 195

	p	r	f1
is from	0.226293	0.30086	0.258303
is about	0.148438	0.112426	0.127946

Table 12: Results of Relation Classification in Epoch 190

	p	r	f1
is from	0.221748	0.297994	0.254279
is about	0.146718	0.112426	0.127303

Some example of the backpropgate errors.(the squared norm of the error matrixes from a randomly picked sentence(same across groups) from the same epoch.)
GroupA:
Target gpyd:0.66459
Agent gpyd:0.302628
DSE gpyd:0.294689
Agent backprop err:0.0490729 0.0326163
DSE backprop err:0.025476 0.0244497
Target backprop err:0.00561583 0.0102002
DSE backprop err:0.00282422 0.00341703
GroupB:
Target gpyd:0.679125

Agent gpyd:0.291735
DSE gpyd:0.299091
Agent backprop err:0.215682 0.134648
DSE backprop err:0.113794 0.110686
Target backprop err:0.0296902 0.0433625
DSE backprop err:0.0105091 0.0112032

4.4 Experiment 4

continue from expr2
setting same as groupB
UPDATE for linear regression
A. from expr2/GroupB
B. UPDATE_RATE = 0.01

Results from GroupB:

Table 13: Results of Entity Extraction

	target		agent		dse	
	prop.	bin.	prop.	bin.	prop.	bin.
p	0.351626	0.353659	0.456176	0.462389	0.454545	0.455497
r	0.0801628	0.203343	0.381767	0.4447	0.0091954	0.574713
f1	0.130561	0.258218	0.415668	0.453372	0.0180261	0.508207

Table 14: Results of Relation Classification in Epoch 195

	p	r	f1
is from	0.125	0.00286533	0.00560224
is about	0.0666667	0.00295858	0.00566572

Table 15: Results of Relation Classification in Epoch 190

	p	r	f1
is from	0.111111	0.00286533	0.00558659
is about	0.333333	0.00295858	0.0058651

4.5 Experiment 6

continue from expr1
setting same as groupA

UPDATE for linear regression
A. from expr1/GroupA
B. UPDATE RATE = 0.01

Results from GroupB:
I also tried the bilinear model, but it doesn't produce better results on rela-

Table 16: Results of Entity Extraction

	target		agent		dse	
	prop.	bin.	prop.	bin.	prop.	bin.
p	0.302778	0.327022	0.423244	0.430736	0.445312	0.453125
r	0.299864	0.445682	0.370819	0.428571	0.0995785	0.131034
f1	0.301314	0.377241	0.395301	0.429651	0.162761	0.203284

Table 17: Results of Relation Classification in Epoch 195

	p	r	f1
is from	1	0	0
is about	1	0	0

Table 18: Results of Relation Classification in Epoch 190

	p	r	f1
is from	0.15	0.0429799	0.0668151
is about	0.156863	0.0236686	0.0411311

tion classification and I haven't found the setting that prevents the gradient explosion in 200 epochs.
My problem now is that there are still other possible neural network structures can be tried. However, each new structure needs some parameter tuning. And I'm not sure if a structure needs more tuning or it simply doesn't work.
I'm now inclined to try a token-level relation classification layer(instead of using a separate logistic regression classifier.) That is, feed the hidden layer representation into a non-linear layer and then having a tensor layer to do token-level relation identification. Ideally, the non-linearity can make sure the hidden layer representation coming from different neural networks fall in the same space.

5 Dec. 8th

5.1 Coreference

The coref in the original system doesn't contain the blogpost authorship resolution. But it can be solved by, either

- add heuristics to the original coref output generated by BBN SERIF.
- combine the output with another coref system.

For approach two, the state-of-the-art coref systems don't solve the author coreference. ([**?**, **?**, **?**])

5.2 Chinese Information Extraction using Neural Nets