Report on relation extraction

Yiqing Hua

November 6, 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 every pair as true pair.

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

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