CS224N PA4: Neural Networks for Named Entity Recognition

Daoying Lin (SUID 06090664) Patrick Manion (SUID)

1 System Design

1.1 Overview of System Implementation Details

1.2 Design Choices

2 Gradient

It can be shown that the expression for $\frac{\partial J(\theta)}{\partial L}$ is:

$$\frac{\partial J(\theta)}{\partial L} = W^T U^T (p_\theta - y) \odot tanh'(Wx + b^{(1)})$$

.

We also generalized the gradient expression for multiple layers of neural network, which is summarized next.

2.1 Gradient Derivation (With Extra Credit)

Let w^l_{jk} denote the weight for connecting the k^{th} neuron in the $(l-1)^{th}$ layer to the j^{th} neuron in the l^{th} layer; b^l_j denote the bias for the j^{th} neuron in the l^{th} layer; a^l_j denote the activation of the j^{th} neuron in the l^{th} layer; z^l_j denote the weighted input to the j^{th} neuron in the l^{th} layer; $h_l(.)$ denote the activation function for the weighted input $\mathbf{z}_l$. Note that $z^l_j = \sum_i w^l_{ji} a^{l-1}_i + b^l_j$ and $a^l_j = h_l(z^l_j)$. Let's define $\delta^l_j = \frac{\partial J}{\partial z^l_j}$, the error of neuron j in layer l. Then it can be easily derived that the following four equations are true for any backpropagation system with any number of hidden layers:

$$\delta^L = \frac{\partial J}{\partial a^L} \odot h'_L(z^L) \tag{1a}$$

$$\delta^l = (W^{l+1})^T \delta^{l+1} \odot h'_l(z^l) \text{ for } l = 1, ..., L-1 \tag{1b}$$

$$\frac{\partial J}{\partial b^l_j} = \delta^l_j \text{ for } l = 1, ..., L \tag{1c}$$

$$\frac{\partial J}{\partial w^l_{jk}} = \delta^l_j (a^{l-1}_k)^T \text{ for } l = 1, ..., L \tag{1d}$$

where $h'_L(z^L) = p_\theta * (1 - p_\theta)$ and $tanh'(x) = 1 - tanh^2(x)$.

For current system, we've three layers: input layer, hidden layer and output layer. The cost function is $J = -ylna^L$. Using the above general system, we can obtain the following:

$$\delta^3 = p_\theta - y \tag{2a}$$

$$\delta^2 = (W^3)^T \delta^3 \odot tanh'(z^2) = U^T(p_\theta - y) \odot tanh'(Wx + b^{(1)}) \tag{2b}$$

$$\delta^1 = (W^2)^T \delta^2 \odot I'(x) = W^T \delta^2 = W^T U^T \delta^3 \odot tanh'(Wx + b^{(1)}) \tag{2c}$$

And

$$\frac{\partial J}{\partial U} = \delta^3 (a^2)^T = tanh(Wx + b^{(1)})(p_\theta - y) \tag{3a}$$

$$\frac{\partial J}{\partial W} = \delta^2 (a^1)^T = LU^T(p_\theta - y) \odot tanh'(Wx + b^{(1)}) \tag{3b}$$

$$\frac{\partial J}{\partial L} = \delta^1 = W^T U^T (p_\theta - y) \odot tanh'(Wx + b^{(1)}) \tag{3c}$$

$$\frac{\partial J}{\partial b^{(2)}} = \delta^3 = p_\theta - y \tag{3d}$$

$$\frac{\partial J}{\partial b^{(1)}} = \delta^2 = U^T(p_\theta - y) \odot tanh'(Wx + b^{(1)}) \tag{3e}$$

2.2 Gradient Check Results

We implemented gradient check and passed the test.
TODO: attach the output maybe?

3 Parameter Tuning

We ran a number of tests to compare the performance of the neural networks. After tuning we found good performance with this benchmark (describe), which achieves 82.42 F1 score on test. To explore the effect of other parameters individually, we use this set of parameters as our base line and vary the parameter we'd like to investigate on. The following are the findings.

3.1 Learning Rate α

3.2 Max Iterations I

3.3 Regularization Value λ

3.4 Hidden Layer Size H

We trained a series of neural networks with single hidden layer while we varied the size of hidden layer which ranges from 100 to 300 with step size of 50. From figure? we observed that the performance is very flat for both training and test set. For single layer neural network, 100 is a good value to choose.

3.5 Window Size C

To study the effect of window size, we varied the value of window from 3 to 15 with step size 2. From figure? We observed there is a substantial performance improvement from 3 to 5. After that, F1 score increases as the window size increases on training. However, F1 score remains relatively flat after 5 on test. This suggests that a window size 5 is a good value to choose and there is probably overfitting on training when window size is greater than 5.

3.6 Fixed vs. Updated Word Vectors

3.7 Randomly Initialized vs. Pre-trained Word Vectors

We also compared the performance of using randomly initialized word vectors and pre-trained word vectors. Based on figure?, pre-training improved the performance dramatically on both training and test data. Does pre-training always help? Intuitively, if words were pertained on relevant corpus this should be the case. But what if words were pre-trained on Wall Street Journal and the problem is to do NER on corpus related to Football or some other not relevant field? Without more experiments we can't make a general conclusion.

4 Error Analysis

We analyzed the error by entity. For each entity, we reported two misclassification errors: False Positive (FP) and False Negative (FN). Take PER as an example, FP means misclassification of non-PER word as PER and FN means misclassification of PER as any non-PER word. Here non-PER word includes LOC, ORG, MISC and O.

4.1 PER

- *False Positive* There are 129 cases where O is misclassified as PER. Among them, mostly are numbers (1, 2, 3, 53.98, 1,627, 1988, etc) and compound adjectives (ex-rebel, Lieutenant-Colonel, newly-signed, over-allotment, soft-spoken, etc.). These misclassification can be avoided if we could enforce some rules. For example, usually numbers won't be a person.

  There are 69 cases where ORG is misclassified as PER. Some examples are: Fed, Duke, Ford, Johnson, Kent, Jones, Lola, Magna. A lot of them are actually organization's that are named after their founders. These would be really hard for the algorithm to distinguish since those words could be either cases.

  There are 38 cases where LOC is misclassified as PER. PATRICK.

  There are 25 cases where MISC is misclassified as PER. One very interesting example is the words "Michael" and "Collins". These two are actually human names. The reason the algorithm misclassified them is because there is a movie named "Michael Collins". For cases like this, it's going to be really hard to make the right prediction.

- *False Negative* The false negative is dominated by misclassifying PER as O, which are mostly non-standard English name. For example: Hondo, Inzamam-ul-Haq, Capelli, Djorkaeff, Hun, Wang, Donghai, Xiao, Sihanouk, etc. This type of error can be reduced by pre-training words on more general text that contains non-standard English name.

4.2 MISC

- *False Positive* The false positive is dominated by misclassification of O as MISC. Again, there are a lot of numbers (13, 40, 1988, etc) and compound adjectives (little-known, 88-year-old, army-backed, etc).

- *False Negative* The false negative is dominated by misclassifying MISC as O. A lot of them are related to sports event like "ENGLISH COUNTY CHAMPIONSHIP", "U.S. Open Tennis Championship", "U.S. Amateur Championship", etc. While the algorithm often correctly classified other words (like U.S. Open) as MISC but failed to classify championship correctly. This is a challenging problem since most of the time "championship" should be O. It's only referring to an entity when in phrase like this. Similarly, MISC is misclassified as LOC because usually those words are indeed location but it's an NER when appears in phrase like "the HONE KONG Open", "the Chicago PMI", "the Berlin Grand Prix", etc. These kinds of error may be reduced by sequence modeling or treating those phrases as if one word.

5 Extra Credit

5.1 Alternative Prepared Word Vectors

5.2 Deeper Neural Networks

Our design and implementation works for any hidden layer architectures. To study the effect of deeper neural networks, we run experiments with two hidden layers and set the size to be the Cartesian combination of $\{50, 100\}$ and $\{50, 150, 300\}$ for the first and second hidden layer, respectively. We also included the performance of baseline model that only has one hidden layer. From ? we can see deeper neural networks indeed improved the performance but only slightly. However, for the same depth, the performance seems not very sensitive to the size of hidden layers. We also implemented 3 and 4 hidden layers of neural networks. Surprisingly, the performance didn't improve as the network gets deeper. More experiments are needed to fully understand the role of the depth of neural networks.