CS224n Fall 2014 Programming Assignment 4

SUNet ID: tzhang54, jiajihu
Name: Tong Zhang, Jiaji Hu

1 System Implementation

1.1 Baseline

In the baseline implementation, we used a Map to store the label of each word. In training stage, we iterated over each training datum and checked if the word is in the map. If the word is not in the map, add it to the map with the corresponding label. If it is in the map but the previous label is inconsistent with the new label, we would update the value to the new label unless the new label is O. For each test datum, our prediction is the label stored in the map. If the word is not in the map, we would predict O.

1.2 Word Vectors

1.2.1 Load From File

Our first option to populate the word vectors is to load the word vectors from file. We read the file in two passes. In the first pass, we obtained the dimension of the word vectors n, and the number of words $|V|$. Then we created an $n \times |V|$ matrix in which each column represented a word vector. In the second pass, we populated the matrix with values from the file.

1.2.2 Random Initialization

Our second option is to randomly initialize the word vectors. We could pass in the word vector dimensions as parameters to create the matrix. Then for each element in the matrix, we generated a random real number between -1 and 1.
The comparision between these two methods of initializing word vectors will be shown in later sections in the report.

1.3 Context Windows

For a given word w_i and window size C, we generated the context windows by concatenating $w_{i-C/2}$ ... $w_{i+C/2}$. If the indices are out of bounds, pad with $< s >$ or $< /s >$ accordingly. Then we converted each word to the word vector to get the final form of the input for the feedforward process.
When looking up a word in the map, we first convert the word to lower case and replaced each digit in the word to DG in order to match the given vocabulary list.

1.4 Feedforward

Our feedforward implementation followed the formula

$$p_\theta(x^(i)) = g(Uf(Wx))$$

The dimensions of each element in the formula are shown in the table 1.

Matrix	Dimensions
x	$(nC + 1) \times 1$
W	$H \times (nC + 1)$
h	$(H + 1) \times 1$
U	$K \times (H + 1)$

Table 1: Feedforward Matrix Dimensions

First, we pad the input vector x with a constant value 1.0 at the end in order to represent the bias term. Then we compute the vector $h = f(Wx)$ and pad a constant 1.0 at the end as well. Our function $f(.) = tanh(.)$. Finally we applied the softmax function g to produce the final probability vector p.
We initialized the matrices W and U by assigning a random value in the range of $[-\epsilon_{init}, \epsilon_{init}]$, including the the bias term.

1.5 Backpropagation

2 Analysis and Plots

3 Error Analysis

4 Extra Credit

4.1 Compare Word Vectors

We downloaded the glove word vector *glob.6B.50d.txt* and converted it to the format similar to the given word vectors and vocabulary list files. We also added the special symbols $UUUNKKK$, $< s >$, and $< /s >$ to make it consistent with the given word vectors. As shown in the results section above, the larger word vectors did help improve our performance.

4.2 Sequence Modeling

We added an interface to support sequence modeling. In sequence modeling mode, the dimension of the following matrices changed, where K is the number of label classes.

Matrix	Original Dimensions	New Dimensions
x	$(nC+1)\times 1$	$(nC+1+K)\times 1$
W	$H\times(nC+1)$	$H\times(nC+1+K)$

Table 2: Matrix Dimensions with Sequence Modeling

We initialized W and x accordingly. Starting from the first iteration, we stored the prediction vector p and input the values to vector x in the next iteration.
The results comparision of sequence modeling are shown in the results section above.