CS59000: Machine Learning for Natural Language Processing

Homework 2

Somaiah Thimmaiah Balekuttira

Due: Nov 21, 2017 on Tuesday

1 Task

Implementation of named entity recognition for the given data set. Implemented one model:Deep Maximum Entropy Markov Model(DMEMM). DMEMM extends MEMM by using a neural network to build the conditional probability.

2 Algorithms

The neural network is trained with the training data. Finally Viterbi algorithm was used for inference. These algorithms are described below :

2.1 Viterbi (Inference)

After training the neural network on the training data. Viterbi inference algorithm is used to find the tags or labels of the words of the sentence. The Viterbi table VT was created using the equation below:

$$VT(col, v) = \max_{u \in tot}(VT(col-1, u) * P(Q|u, xcol))$$

where,col represents the column position of word k, xcol is the word in column col, Q is the tag for which probability is being calculated, u is the tag and tot is the total set of tags.

Initialization : The first column of the Viterbi table built for each sentence represents the start word
The row corresponding to the start tag of this column is made to be 1 which represents a probability of 1 and the rest of the entries of this column are made into 0.

For each word in the test sentence, P(Q—u,xcol) was calculated from the deep memm neural network. This means that the probability table is calculated from the output of the deep memm neural network for the current word and every tag (label). For u ranging from start tag to end tag. The last column is for the end word, which always has the end label. So the probability of all

labels was set to 0 except the end label. This way the probability table is created using the neural network. The values from the previous column of the current word being investigated from the viterbi table is multiplied with every row of the probability table. The entries which corresponds to max value in every column is picked and updated for the current words column in the viterbi table. This way the viterbi table is generated.

2.2 Three layer neural network

The neural network is three layer. The input nodes are equal to the size of the input embedding and the output nodes are equal to the size of the tag (label) vectors. This implemented neural network was found to give best results with the number of neurons in the hidden layer equal to 400 and the learning rate to be 0.001. These hyperparameters were altered for best performance with the help of the validation set.

Tanh function is used as the activation function for the neural network.

Softmax function is used in the output layer. It is applied to all slices and will rescale them so that the elements lie in the range (0, 1) and sum to 1 . This is necessary as we expect probabilities for each tag (label) for the current word in the neural network. The Loss Function used in the neural network is the MSELoss: This creates a criterion that measures the mean squared error between n elements in the input x and target y. This function is given by:

$$loss(X, Y) = 1/n \sum (Xi - Yi)^2$$

Output: The neural network outputs a vector of size equal to the number of tag (labels) . This vector essentially contains the probability for the tags (labels) being the actual label for the current word in the network given its previous tag. This prediction is then compared with the one-hot representation of the actual tag label for the particular word to calculate the loss. This loss is backward propagated using Pytorch functions.

3 Experimental Setting

The training data consists of sentences and tags (labels) for every word in the sentences. This training data was used to train the neural network. The validation set data was used to find the best hyper parameter values. Finally the recall, accuracy and f-score are calculated on the test data.

3.1 Feature extraction

Every word in the dataset is represented by a vector of size 400 initialised randomly. Every tag (label) is represented by a one-hot representation. Two new words are added , the first being the start word and the second being the end word. These two words are added to every sentence while using Viterbi algorithm for inference. Two new tags , the start tag which is the tag for the start word and the end tag which is the tag for the end word is introduced. Every sentence in the training data was parsed to extract individual words. The corresponding word embedding which

represents these words are concatenated with the embedding of the tag (label) of the previous word. This embedding is the final feature vector which is the input to the neural network.

3.2 BackTracking For Inference

Backtracking is a technique used to find the final labels or tags of the words of the test sentence. The L (i.e. the label) for which

$$VT(k-1, L) * P(v|L, xk)$$

was maximum was saved in the in the backtrack table BT. The equation is given below:

$$BT(k, v) = L$$

Hence, the backtrack table stores the label L of the previous word, with prior knowledge that the label of the current word is v.

The following is the sequence of steps used in the backtracking: We take the help of two new words the start word and the end word. The start word has the start tag whereas the end word has the end tag. Let the number of words in the test sentence be n. To find the tags or labels of the words of the test sentence, we start at the label Q in the last column of the backtrack table. Which corresponds to the max value in that column.

$$Q = BT(n+1, endlabel)$$

This is the label for the n^{th} word. For rest of the words, we will calculate the label Q' as follows:

$$\text{while} \quad (n \quad <= \quad 2):$$

$$\begin{aligned} Q' &= B(n, Q) \\ Q &= Q' \\ n &= n-1 \end{aligned}$$

This way all the labels of the words of the sentence are predicted.

4 Results

4.1 Neural network + Viterbi Inference

Table 1: Results

Model	Precision	Recall	F1 score
DMEMM	67.50	66.41	66.95