CS59000: Machine Learning for Natural Language Processing

Homework 2

Mohit Gupta

Due: Nov 21, 2017 on Tuesday

1 Task

Implement Deep Maximum Entropy Markov Model (DMEMM) in combination with the implementation of Viterbi Algorithm used for inference at test time for named entity recognition.

2 Algorithms

2.1 Deep Maximum Entropy Markov Model (DMEMM)

- I implemented a 3-layer neural network with 2 versions of input vectors with below features:
  - current word cancatenated with label of previous word.
  - current word concatenated with previous word and label of previous word.

  The above mentioned features were used for each word with output of the neural network corresponding to the current word. When I used the features mentioned in the 2nd point, I got f1 score higher than those mentioned in the 1st point.
- For the hidden layer, I used tanh as activation function which gave me the highest f1 score.
- For the output layer, I used softmax so that the probabilities of the output labels is between 0 and 1.
- I tried Mean squared error (MSE) loss and cross entropy loss. For the final model, MSE loss was selected based on tuning with the validation data set since it gave me higher f1 scores.

2.2 Viterbi (Dynamic Programming) Algorithm for Inference

During test time, I used Viterbi algorithm to infer the named entity tags.

- I created dynamic programming matrix dp using equation:

$$dp(k, v) = \max_{u \in S}(dp(k-1, u) * P(v|u, x_k))$$

  here, v is the tag for which value is being calculated, u is the tag of the previous word and belongs to the set S (set of all tags), xk is the current word with k representing kth word in the sentence.

- If there are n words in a sentence, then my dp matrix has n+1 columns with each word representing corresponding column from 1 to n. The 0th column represents start tag. Value of dp[0][0] is 1 and rest all values in 0th column are initialized to 0.

- For each word, I calculate the probability of the tag of the current word given the previous tag and the current word from the neural network. I then use the dp recurrence equation as given above to find the final value of tag of the current word in the dp matrix.

- Finally, I find the max value of the tag for the nth word from the corresponding column from dp matrix and backtrack till the first word column to finally get all labels. Thus, I predict the named entity tags of the sentence.

3 Experimental Setting

3.1 Training, Validation and Test Data

- Preprocessed Data split into Training, Validation and Test data set was provided.

- The neural network was trained on provided Training Data. Validation data set was used to tune the hyperparameters namely learning rate, coefficient of l2 regularizer, dimension of word embedding, number of epochs for training the neural network, number of neurons in the hidden layer and finding the better optimisation method i.e. Stochastic Gradient Descent or Adam.

3.2 Feature Selection

- A simple neural network is trained with randomly initialized word embeddings to represent a word and bit mapping embeddings to represent label of previous word. The dimension of bit mapping embeddings is equal to the number of labels in the given data set which is equal to 127.

- The word embeddings currently being used after tuning with the validation data set are stored in word_embeddings.pt. These embeddings are of dimension 300 and were randomly initialised with each element having value greater than equal to 0 and less than 1.

- The current word vector for which label needs to be predicted is concatenated with the label of the previous word and passed as input to the neural network.

- To improve performance, more features are added i.e. the word embedding of the previous word.
- The previous word embedding, the current word embedding and the bit mapping embedding of the label of the previous word are concatenated and passed as input to the neural network.

3.3 Hyperparameter Selection

- I trained the neural network using 2 optimizer functions namely Stochastic Gradient Descent (SGD) and Adam. Learning rate for Stochastic gradient descent was varied between 0.001, 0.01 and 0.1. For Adam, learning rate was varied between 0.00005 and 0.0005. Based on the speed of training and fastness of convergence with good scores(91 f1 score) on the Validation data set, Adam was selected for the final model which was then run on the test data set.
- The neural network was trained both with and without adding a l2 regularizer. Pytorch has an implementation of l2 regularizer in the optimizers. An argument weight_decay needs to be provided in the function with the coefficient value for the same. The coefficient of the l2 regularizer was kept at 1e-5. Adding the l2 regularizer slowed the training requiring more number of epochs to obtain comparable results to those without using regularizer.
- The neural network was trained using 1 hidden layer. The number of hidden nodes were varied from 200 to 1000. Finally, the value of 450 was selected since more number of hidden nodes slowed the training of the model. The value of 450 was giving optimal results. I observed that the optimal value of the number of hidden nodes is around the mean of number of input nodes and number of output nodes. I read about discussions around the same on many blogs discussing optimal value of number of hidden nodes for simple neural networks.
- The dimension of word embeddings is fixed at 300 after tuning it using the validation data set.
- The number of epochs is fixed at 2600 after tuning it using validation data set.
- The trained neural network model is saved in parameters.pt and is loaded for inference.

4 Results

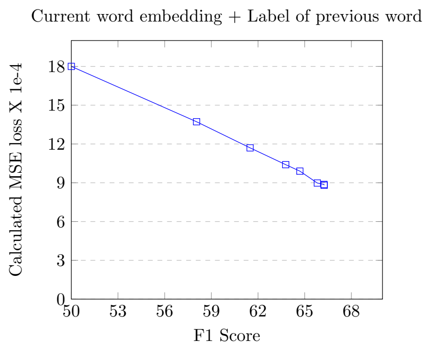
4.1 Tabular Results

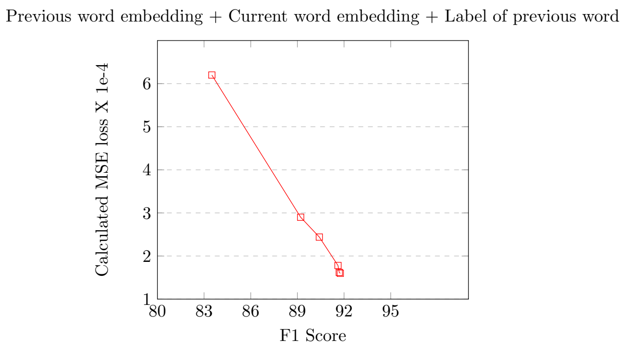
Results on Validation Data set and Test Data set are shown in the below tables respectively.

Results - Validation	Data	Set		
Features	Epochs	Precision	Recall	F1 score
Current word embedding + Label of previous word	2000	66.38	66.13	66.25
Previous word embedding + Current word embedding + Label of previous word	2600	91.83	91.4	91.61

Results - Test	Data	Set		
Features	Epochs	Precision	Recall	F1 score
Current word embedding + Label of previous word	2000	65.47	63.55	64.5
Previous word embedding + Current word embedding + Label of previous word	2600	90.01	88.58	89.29

4.2 Graphs showing Relation between MSE loss on Training Data set with F1 scores on Validation Data set





4.3 Key Points

- The MSE loss function plateaus at around 0.00088 with f1 score of 66.25 on validation data set when concatenation of current word embedding and label of previous word is used as input vector to the neural network.
- The MSE loss function plateaus at around 0.00017 with f1 score of 91.61 on validation data when concatenation of previous word embedding, current word embedding and label of previous word is used as input vector to the neural network.