Project 1: CRFs for NER

Abstract

In this assignment, we look at the problem of Named Entity Recognition (NER), which is a sequence labeling task. We approach the problem using chain CRF model. Further, we compare the CRF model to structured SVM, in terms of F1 score, learning curve, training time and variation across multiple runs.

1 Problem statement

Named Entity Recognition (NER) is a natural language processing task that attempts to find named entities (persons, locations, organizations, etc.) within a text. Since the tag of one word depends on neighboring words (and their tags), the problem is modeled as a sequence labeling task, wherein a sentence is considered as an input, and the goal is to produce the tags of all words jointly, so as to maximize accuracy.

Evaluation Metric: Since most of the tags are 'O', using a simple token-wise accuracy is not representative of the actual performance of the model. So, we compute chunks using the BIO tags to compute predicted entities, and compare them to the ground truth entities, to calculate an F_1 score.

2 Basic CRF model

2.1 Implementation details

Forward-backward algorithm: We use forward-backward algorithm to compute marginal probabilities required by stochastic gradient descent on log likelihood. For the forward pass, the scores of the first word are set to the emission potentials, and for backward pass, the scores of the last word are initialized to 1 (or 0 in the log space).

Gradient descent: Stochastic gradient descent was used with constant step size. The value of the step size was determined to be 0.2, using validation. In each epoch, the sentences in the training corpus were shuffled, and one sentence was used per step of gradient descent.

Initial and Transition potentials: Initial and transition potentials were hard-coded to prevent illegal tag sequences, by disallowing an 'I' tag to be at the start of the sentence, or following an 'O' tag or a 'B' tag from a different category.

Viterbi decoding: In order to reuse the Viterbi decoding implementation for HMMs which takes a precomputed table of emission potentials for all (tag, word) pairs, a wrapper was written that computes emission potentials for each sentence. Specifically, given a sentence, the Viterbi algorithm will only require emission potentials of (tag, word) pairs for words that appear in the given sentence. However, since in CRFs, the emission features are dependent on the position of the word, using an indexer that maps a word literal to an index in the emission table doesn't work if the sentence has duplicates. Therefore, in the wrapper, an auxiliary sentence was created where each word token is identical to the word position, and a word indexer maps the word tokens ('0', '1', . . ., 'N') to indexes (0, 1, . . ., N) respectively, where N is the length of the sentence. The corresponding emission table (in the log space) is computed by multiplying the learnt feature vector with the emission features for the words.

Results: The resulting CRF model achieves an F1 score of 88.29 on the development set.

3 Extensions: Structured SVM model

3.1 Implementation details

Loss-augmented Viterbi decoding: To compute the most violated constraint, Viterbi algorithm was extended so that for every word position, the score of each incorrect tag was incremented by 1. The resulting dynamic program returns the correct tag sequence if the scores of the correct tags at each step are larger than the next highest score by at least a margin of 1. Otherwise, it returns the most violated sequence of tags according to the structured SVM constraint set. The output is then used to compute the gradient for the weight vector.

Gradient descent: As in CRF model, we use one sentence per gradient step for stochastic gradient descent with a fixed step size. The values of the step size and the regularization parameter were found to be 0.5 and 0.00001 respectively using validation on the development set.

Results: The structured SVM model achieves an F1 score of 83.16 on the development set.

3.2 Comparison of CRF and structured SVM models

In the following, we compare the two models on various different parameters.

Accuracy vs training iterations: The graph below shows the F1 scores of the models plotted as a function of the number of epochs.

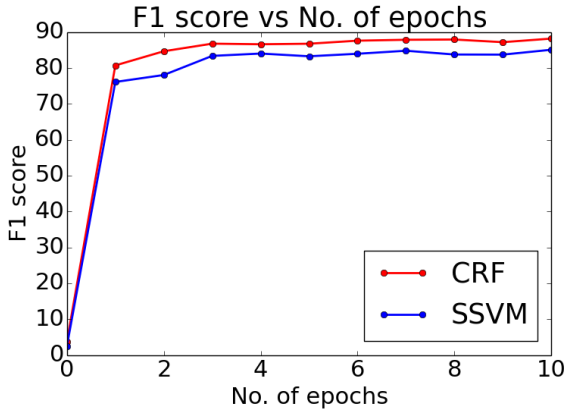

Figure 1: F1 score comparison for CRF and structured SVM as a function of the number of epochs.

As can be seen from the figure above, both the models are fairly similar in terms of converging to their respective F1 scores.

Accuracy vs number of samples: The graph below shows the F1 scores of the models plotted as a function of the number of training samples.

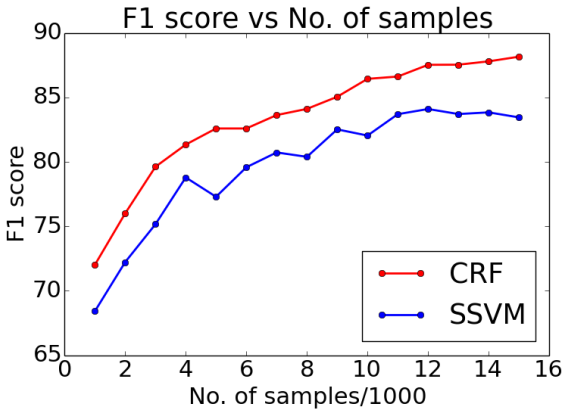

Figure 2: F1 score comparison for CRF and structured SVM as a function of the number of training samples.

Again, we see that the convergence behavior is similar for both curves.

Variation across multiple runs: An interesting observation can be made from the graph above – while the curve for CRF is monotonically increasing, the curve for structured SVM does not. This suggests that the performance of structured SVM varies more across multiple runs. To test this hypothesis, we run both models with best parameters for 5 independent runs, and compute the mean and standard deviation of the F1 scores.

For CRF, the mean F1 score was obtained to be 88.29, and the standard deviation was found to be 0.18. For structured SVM, the mean F1 score was obtained to be 83.16, and the standard deviation was found to be 1.96. Thus, we see that it is indeed true that different runs of SVM are likely to generate more varied results, possibly because it is more sensitive to the stochasticity of SGD. One way to get around this problem is to increase the batch size to have better estimates of gradients at every step, at the expense of increasing the training time.

Training time: The mean training time of CRF was found to be about 57 minutes, while that of structured SVM was found to be about 28 minutes. Therefore, training structured SVM is about

twice as fast as training CRF. This factor of two probably comes from the fact that in training CRF, every gradient computation needs to perform a forward and backward pass, while in training structured SVM, only a forward pass is required in loss-augmented Viterbi algorithm.

4 Conclusions

We implemented a chain CRF model for NER tagging of English sentences, and compared the performance of the model to that of structured SVM. We observe that while structured SVM is twice as fast to train compared to CRF model, the F1 score generated by CRF model is about 3.5% better in relative terms. Further, CRF appears to be less sensitive to stochasticity in gradient descent.