(**20 points**) Create a baseline that achieves more than or equal to 60% F1 on the development set.

The constrained version is shown in the Jupyter Notebook (HW4_JiangshanLuo.ipynb) Part 1.

**(10 points)** You will get 10 points for at least an attempt on the unconstrained version beyond what is required for the baseline.

The unconstrained version is shown in the Jupyter Notebook (HW4_JiangshanLuo.ipynb) Part 2.

Demonstrate that you have thought about the problem carefully, and come up with solutions both for the baseline **(20 points)** and for your unconstrained version **(10 points)** in your write up. Extra credit **(5 points)** for top-3 performance on the test set.

The first thing I designed for my models are some local knowledge features since they can indicate the label of the word in some degrees. For instance, I put checks on whether the word contains digits, initial letters, special signs, etc.

More importantly, one feature that significantly influences the model performance is how far the model looks forward/backward, or how many neighbors of the current word the model considers during the input process. However, increasing the sliding windows of neighbors tremendously increases the burden of training the model so there is a trade-off here.

And for my unconstrained version, I tried the feed-forward network with google pre-trained Word2Vec (word embedding). As a result, the performance of it is similar to the constrained model, which is lower than my expectation. One obvious reason is that the input dimension is relatively high (with Word2Vec ), thus my current FFN might be too simple to handle. If time permitted, larger hidden layers, different optimizers as well as training epochs would be used to find a better version of the FNN model.