1. The skeleton code we have given you gets about 49% F1 (on development set) right o ut of the box. (20 points) Create a baseline that achieves more than or equal to 6 0% F1 on the development set.

I read the paper 'Named Entity Recognition: Exploring Features' and tried the feat ures from local information about a current token. Base on that I implement getsha pe() function and choose the features in the red box to put into the X_train.

Features	Dev	Test
CoNLL-2003		
w_0	25.24%	22.04%
w_0 + suffixes and prefixes	87.41%	78.59%
$w_0 + s_0$	86.70%	79.16%
$w_0 + s_{-1}, s_0, s_1,$ $s_{-1}\&s_0, s_0\&s_1, s_{-1}\&s_0\&s_1$	87.67%	81.37%
All Local Features	88.91%	82.89%

w represents token and p represents POS tags. Subscript index stands for the token which clustering label is used. -1 stands for the previous token, +1 stands for t he next token; 0 stands for the current token. And after training, I find that the result is not as good as I think. It's just 50 for the F1(The result is shown belo w).

```
processed 52923 tokens with 4351 phrases; found: 5247 phrases; correct: 2441.
accuracy:  93.61%; precision:  46.52%; recall:  56.10%; FB1:  50.86
              LOC: precision:  65.08%; recall:  51.32%; FB1:  57.39  776
             MISC: precision:  15.42%; recall:  35.28%; FB1:  21.46  1018
              ORG: precision:  47.11%; recall:  61.82%; FB1:  53.47  2231
              PER: precision:  59.57%; recall:  59.57%; FB1:  59.57  1222
```

Base on the current result I tried to add more features. The paper says that POS t ag works pretty well in this situation, so I tried to add this feature to my progr am and it turns out that it did improve the performance but the value of F1 still does not reach 60 it's about 56.

And this time I choose to change the model I used for the training part. I tried m any methods from this link(https://scikit-learn.org/stable/modules/classes.html# module-sklearn.linear_model) and some of them have totally different input data f ormat and others required dense data which will use too much memory. After compari ng, I chose logistic regression as my training model.

```
processed 52923 tokens with 4351 phrases; found: 4918 phrases; correct: 2914.
accuracy:  95.28%; precision:  59.25%; recall:  66.97%; FB1:  62.88
              LOC: precision:  56.52%; recall:  76.63%; FB1:  65.06  1334
             MISC: precision:  31.86%; recall:  37.30%; FB1:  34.37  521
              ORG: precision:  61.39%; recall:  65.00%; FB1:  63.14  1800
              PER: precision:  70.39%; recall:  72.75%; FB1:  71.55  1263
```

Finally, I got 62.88 for the F1 score.

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

For this part, I kept the logistic regression as my training model and add new clu ster features.

Features	Dev	Test
CoNLL-2003		
p_0	45.63%	43.98%
$w_0 + p_0$	83.07%	73.42%
b_0	80.98%	75.51%
$w_0 + b_0$	89.35%	82.17%
c_0	67.47%	64.06%
$w_0 + c_0$	86.47%	79.29%
l_0	45.20%	44.24%
$w_0 + l_0$	82.28%	72.63%
g_0	79.90%	76.72%
$w_0 + g_0$	88.36%	81.98%
$b_0 + c_0 + l_0$	86.40%	80.76%
$b_0 + c_0 + l_0 + g_0 + p_0$	89.26%	84.66%
$w_0 + b_0 + c_0 + l_0 + g_0 + p_0$	90.87%	87.00%

From the paper I notice that Brown clusters work quite well so I choose to add thi s feature and added the POS tags which were already used in the first question. I didn't use the pervious and next word for the current word in this question. Becau se Brown clusters will cost a lot of time and I think it's already enough to get 6 0 F1 scores base on this paper. And that is the result I got.

```
processed 52923 tokens with 4351 phrases; found: 4969 phrases; correct: 2851.
accuracy:  95.12%; precision:  57.38%; recall:  65.53%; FB1:  61.18
              LOC: precision:  55.62%; recall:  76.42%; FB1:  64.38  1352
             MISC: precision:  29.98%; recall:  30.79%; FB1:  30.38  457
              ORG: precision:  55.33%; recall:  63.47%; FB1:  59.12  1950
              PER: precision:  72.98%; recall:  72.26%; FB1:  72.62  1210
```