

## รายงาน

### Data preprocessing (load\_data function)

```
def load_data(file_path):
    data = open(file_path).read()
    list_all = data.split('\n')
    sentence_in_list_all = []
    sentence = []
    answer_isus = []
    Y = []
    X = []
    # print(list_all)
    for i in list_all:
        if i == '':
            sentence_in_list_all.append(sentence)
            sentence = []
        else:
            sentence.append(i)
    for sentences in sentence_in_list_all:
        answer = []
        keep_all_word_and_ner_each_sentence = []
        for len_word in range(len(sentences)):
            word_and_ner = []
            word_and_ner_split = sentences[len_word].split("\t")
            word_and_ner.append(word_and_ner_split[1][2:])
            word_and_ner.append(word_and_ner_split[0])
            keep_all_word_and_ner_each_sentence.append(word_and_ner)
        # print(keep_all_word_and_ner_each_sentence)

        replaced = ['x' if x[0]==' ' else x for x in keep_all_word_and_ner_each_sentence]
        # print(replaced)
        keep = []
        for len_word_and_ner in range(len(replaced)):
            if replaced[len_word_and_ner] != 'x':
                if len_word_and_ner == len(replaced)-1:
                    list_answer = []
                    # print("1: add",replaced[len_word_and_ner][1])
                    keep.append(replaced[len_word_and_ner][1])
                    list_answer.append(replaced[len_word_and_ner][0])
                    list_answer.append(''.join(keep))
                    answer.append(tuple(list_answer))
                    # print('KEEP:',keep)
                elif replaced[len_word_and_ner][0] == replaced[len_word_and_ner+1][0]:
                    # print("2: add",replaced[len_word_and_ner][1])
                    keep.append(replaced[len_word_and_ner][1])
                    # print('KEEP:',keep)
                elif replaced[len_word_and_ner][0] != replaced[len_word_and_ner+1][0]:
                    list_answer = []
                    # print("3: add",replaced[len_word_and_ner][1])
                    keep.append(replaced[len_word_and_ner][1])
                    # print('KEEP:',keep)
                    list_answer.append(replaced[len_word_and_ner][0])
                    list_answer.append(''.join(keep))
                    answer.append(tuple(list_answer))
                    keep = []
            else:
```

```

    # print(4)
    continue
# print(answer)
answer_isus.append(answer) #เอาไป evaluate
list_tempX = []
list_tempY = []
for i in answer:
    list_tempX.append(i[1])
    list_tempY.append(i[0])
X.append(list_tempX)
# X.append(pos_tag(list_tempX, corpus='orchid_ud'))
Y.append(list_tempY)
return (X,Y,answer_isus)

```

```

X_train, Y_train, evaluate_train = load_data('/content/drive/MyDrive/CONTEST 3/Contest 3 - NER/train_auto_tok.tsv')
X_dev, Y_dev, evaluate_dev = load_data('/content/drive/MyDrive/CONTEST 3/Contest 3 - NER/dev_auto_tok.tsv')

```

เริ่มจากเอา train\_auto\_tok.tsv มาเป็น data ของการเทรนโมเดลโดยที่ใช้ฟังก์ชัน  
load\_data ทำกระบวนการภายใน

โดยใน load\_data function จะทำการดึงตัวที่มี NER เหมือนกันมาติดกันเท่านั้น  
และผมได้ทำการใส่ part of speech โดยใช้ pythainlp จะได้หน้าตาของ X\_train (1 sentence)  
เป็นรูปแบบประมาณนี้

```
1 X_train[-5]
```

```

[('เวลาประมาณ 09.20 น.', 'NOUN'),
 ('นาย', 'NOUN'),
 ('รองพล เจริญพันธ์', 'PROPN'),
 ('ทำเนียบรัฐบาล', 'NOUN'),
 ('รัฐมนตรี', 'NOUN'),
 ('สำนักนายกรัฐมนตรี', 'NOUN'),
 ('นาย', 'NOUN'),
 ('เนวิน ชิดชอบ', 'PROPN'),
 ('สำนักนายกรัฐมนตรี', 'PROPN'),
 ('คณะปฏิรูปา', 'NOUN'),
 ('กองบัญชาการ', 'NOUN'),
 ('กองทัพบก', 'NOUN')]

```

โดยเราได้เริ่มการทดลองโดยแบ่งตามfeatureที่จะนำมาใช้แล้วให้model predictแล้วevaluateกับ test ซึ่งเราจะได้overall F1-scoreมา

1. เริ่มจากการดึง data จาก train มาแล้วทำ CRF model แล้วจึงลอง evaluate กับ test

```
{
  "0 Overall F1": 0.699843097782646,
  "0 Overall P": 0.697262859839547,
  "0 Overall R": 0.7024425031199857,
  "1 ORG F1": 0.6794425087108014,
  "2 PER F1": 0.7703606985474131,
  "3 MEA F1": 0.7369648509078793,
  "4 LOC F1": 0.5372353081132525,
  "5 TTL F1": 0.8452950558213717,
  "6 DTM F1": 0.7843438219493478,
  "7 NUM F1": 0.5684591052869409,
  "8 DES F1": 0.7352941176470589,
  "9 MISC F1": 0,
  "10 TRM F1": 0.3026315789473684,
  "11 BRN F1": 0
}
```

2. ทำfeature POS โดยใช้ library ของ pythainlp โดยTag words with corpus *orchid\_udc* แล้วนำมา ทำCRF model แล้วจึงลองevaluateกับtest

```
{
  "0 Overall F1": 0.8004037645102872,
  "0 Overall P": 0.7997152349311818,
  "0 Overall R": 0.8010934807155168,
  "1 ORG F1": 0.782608695652174,
  "2 PER F1": 0.856086079354405,
  "3 MEA F1": 0.8204288754296939,
  "4 LOC F1": 0.7322761194029851,
  "5 TTL F1": 0.9761801645734084,
  "6 DTM F1": 0.7296819787985865,
  "7 NUM F1": 0.6117290192113245,
  "8 DES F1": 0.9408396946564884,
  "9 MISC F1": 0,
  "10 TRM F1": 0.5921787709497206,
  "11 BRN F1": 0.10714285714285714
}
```

3.ทำfeature word window (bigram)+POSแล้วนำมาทำCRF model แล้วจึงลองevaluate กับ test

```
"0 Overall F1": 0.6491300501327042,  
"0 Overall P": 0.6442662295849675,  
"0 Overall R": 0.6540678671183218,  
"1 ORG F1": 0.6904309252217997,  
"2 PER F1": 0.7181512359221881,  
"3 MEA F1": 0.6897302001740645,  
"4 LOC F1": 0.6143633071816537,  
"5 TTL F1": 0.650156561343581,  
"6 DTM F1": 0.5508390918065154,  
"7 NUM F1": 0.6044997039668444,  
"8 DES F1": 0.6746586746586747,  
"9 MISC F1": 0,  
"10 TRM F1": 0.20155038759689922,  
"11 BRN F1": 0.06547619047619047
```

4.ทำpos tag window feature (bigram) +posแล้วนำมาทำCRF model แล้วจึงลองevaluate กับ test

```
{  
  "0 Overall F1": 0.5682842904603702,  
  "0 Overall P": 0.5673082617708025,  
  "0 Overall R": 0.5692636833660188,  
  "1 ORG F1": 0.6250426475605596,  
  "2 PER F1": 0.7543358109395846,  
  "3 MEA F1": 0.658382286488929,  
  "4 LOC F1": 0.5745752045311517,  
  "5 TTL F1": 0.9347280334728033,  
  "6 DTM F1": 0.5457364341085271,  
  "7 NUM F1": 0.6233453670276775,  
  "8 DES F1": 0.8149210903873745,  
  "9 MISC F1": 0,  
  "10 TRM F1": 0.4833333333333333,  
  "11 BRN F1": 0.013214285714285713  
}
```

5.ทำpos conjunctive feature(bigram)+posแล้วนำมาทำCRF model แล้วจึงลองevaluate กับ test

```
{
  "0 Overall F1": 0.6671212166128124,
  "0 Overall P": 0.665562522181474,
  "0 Overall R": 0.6686872288583824,
  "1 ORG F1": 0.7209228824273073,
  "2 PER F1": 0.7669250645994833,
  "3 MEA F1": 0.7295898048586221,
  "4 LOC F1": 0.6692139737991266,
  "5 TTL F1": 0.8082679971489665,
  "6 DTM F1": 0.5055679287305123,
  "7 NUM F1": 0.5651447661469933,
  "8 DES F1": 0.8585237258347979,
  "9 MISC F1": 0,
  "10 TRM F1": 0.6384615384615385,
  "11 BRN F1": 0.023369036027263874
}
```

6. feature word window (bigram)+ pos tag window(bigram) +posแล้วนำมาทำCRF model แล้วจึงลองevaluate กับtest

```
{
  "0 Overall F1": 0.6165351548698622,
  "0 Overall P": 0.6109587442376145,
  "0 Overall R": 0.6222142984489214,
  "1 ORG F1": 0.6420260095824777,
  "2 PER F1": 0.6935186498369026,
  "3 MEA F1": 0.6679902934039267,
  "4 LOC F1": 0.5470473718364699,
  "5 TTL F1": 0.5700748129675811,
  "6 DTM F1": 0.5801819052178075,
  "7 NUM F1": 0.6268656716417911,
  "8 DES F1": 0.7235272858691724,
  "9 MISC F1": 0,
  "10 TRM F1": 0.6515837104072398,
  "11 BRN F1": 0.0409165302782324
}
```

7.ทำการevaluateโดยใช้ featureที่ทำทั้งหมดแล้วนำมาทำCRF model แล้วจึงลองevaluate กับ test

```
{
  "0 Overall F1": 0.5656274792113537,
  "0 Overall P": 0.5593978844589097,
  "0 Overall R": 0.5719973851548107,
  "1 ORG F1": 0.574033552151714,
  "2 PER F1": 0.6086850152905199,
  "3 MEA F1": 0.6346545866364666,
  "4 LOC F1": 0.45241581259150804,
  "5 TTL F1": 0.4411481410132922,
  "6 DTM F1": 0.5073417721518988,
  "7 NUM F1": 0.5958806818181818,
  "8 DES F1": 0.7453560371517027,
  "9 MISC F1": 0,
  "10 TRM F1": 0.7048458149779735,
  "11 BRN F1": 0.10526315789473682
}
```

จาก F1 score ที่ลองทุกfeature แล้วพบว่าทำfeature POS โดยใช้ library ของ pythainlp โดย Tag words with corpus *orchid\_ud* แล้วนำมาทำCRF model แล้วจึงลองevaluate กับ test จะได้ F1 score สูงสุดซึ่งก็คือ 0.8