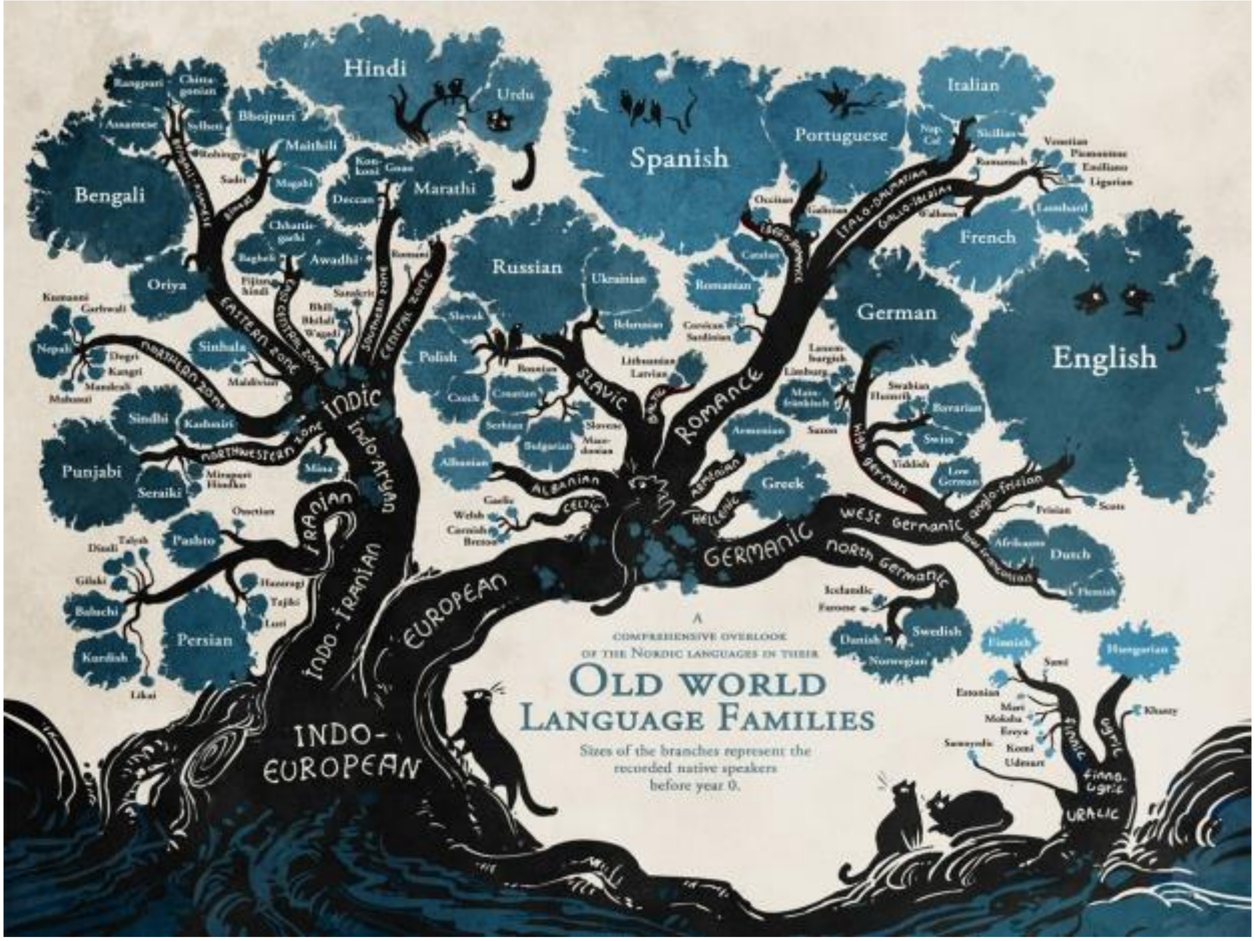


EXERCISE-7 Word Boundary Segmentation with CRF

19 Oct 2019

Phyo Thu Htet, Information Science Student, Software Lab, UTYCC

It is such a great pleasure to explore, think, exploit and code for this exercise.

I explored a lot things that are invaluable things that deliver taste of wonder, happiness, problem solving skills and information that have never touched or imagined . So I want to describe what I have done in particular order.

Although saying particular, actually it is not a linear recipe but an iterative one.

Data Preparation:

The data is prepared into **character-level breaking and syllable-level breaking.**

Python program is used for splitting the data into character level.(List is main data structure for this program)

Perl Programming with regx is used for syllable breaking.

The script concept is checking "က-အ with proceeding character and followed by the characters that are written after the consonants of Burmese language " up to n times. {0,} is use d for this purpose.Otherwise the syllable is take as others.

In approach, the concept need to change for the following three scenarios.

(1) ေ proceeding character is written first according to the standard and style of Burmese Language and Unicode Burmese.

For example, ေက is in က ေ order in processing the texts in back-end. (for i in : print(i)). So there is no proceeding characters.

(2) ္ can be range from က-အ and also have difficulties in adding to the all characters group.

(3) There are some cases that I found.

Eg.

ဦ is not the same is ဥ+ ီ .

ဧ can be preceded by ေ.

The last modified script is

$line =~ s/([က-အ|ဥ|ဦ]([က-အ][်္]|္[က-အ]|[ါ-ှ]){0,}|.)/$1\n/g;

Tagging

List of segmentation tags

- < The first syllable/ character in a word
- > The second last syllable/character in a word
- + Represents both < and >
- - Others
- | Final syllable/ character in a word

Number of tags	Tag set
2	-|
3	<-|
4	<>-|
5	<>+-|

References:

WIn Pa Pa, Ye Kyaw Thu, Andrew Finch and Elenro Smnita, Word Boundary Identification for Myanmar Text using Conditional Random Field

Conditional Random Field Latin Word Segmenter Dylan Rhodes (dylanr) December 8, 2013
https://nlp.stanford.edu/courses/cs224n/2013/reports/dylanr.pdf

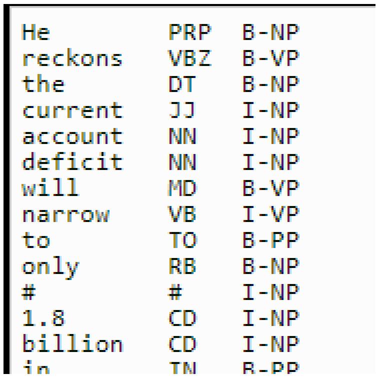
Python code is used for this section.(File, List, Exception Handling)

CRF Model

The Conditional Random Field (CRF) used is an C++ implementation namely crfsuite. https://taku910.github.io/crfpp/ . It is an open source statistical learning model. CRF are can be categorized into the type of discriminative undirectedprobabilistic model. And there was huge amount of success in word segmentation like Arabic, Latin and Chinese. According to Standford University report, the university used that model for the implementation in their researches. https://nlp.stanford.edu/courses/cs224n/2013/reports/dylanr.pdf

CRFs is useful for the process of NLP like POS tagging, name entity recognition, word boundary identification, Text Chunking etc. It possess such a strong popularity for Natural Language Processing.

In using this toolkit , we need to pepare the data into format that the model defines. The data format is shown below.



```
He        PRP  B-NP
reckons   VBZ  B-VP
the       DT   B-NP
current   JJ   I-NP
account   NN   I-NP
deficit   NN   I-NP
will      MD   B-VP
narrow    VB   I-VP
to        TO   B-PP
only      RB   B-NP
#         #    I-NP
1.8       CD   I-NP
billion   CD   I-NP
in        IN   B-PP
```

The second is the template. Unigram and Bigram template type are provided for implementation.

Unigram template: first character, **'U'**

Bigram template: first character, **'B'**

And

```
% crf_learn -f 3 -c 1.5 template_file train_file model_file
```

```
% crf_test -m model_file test_files
```

commands are used for training and testing section.

Evaluation

Firstly, Confusion Matrix is calculated first. The other scoring matrix like (Accuracy, Recall etc) are evaluated getting the data from confusion matrix. In creating matrix in python, the concept is not the same like other programming languages.

First Way:confusion_matrix = []#Please Take Notem=[]for i in unique_elements: m.append(0)   confusion_matrix.append(m)

Second Way:confusion_matrix = [[0] * len(unique_elements)]*len(unique_elements)

The first two ways have the feature of aliasing in creating matrix they will refer the same elements

confusion_matrix[unique_elements.index(data[-1])][unique_elements.index(data[-2])] += 1So the above statement will plus 1 to all

E.g: [[2, 1, 1], [2, 1, 1], [2, 1, 1]]

The solution to these matrix dilemmas is solved by using other style of matrix creation. The solution is shown in the code.

Accuracy

(sumi((TPi+TNi) / (TPi+TNi+FPi+FNi))) / number of unique elements in confusion matrix

Precision

(sumi((TPi) / (TPi+TNi))) / number of unique elements in confusion matrix

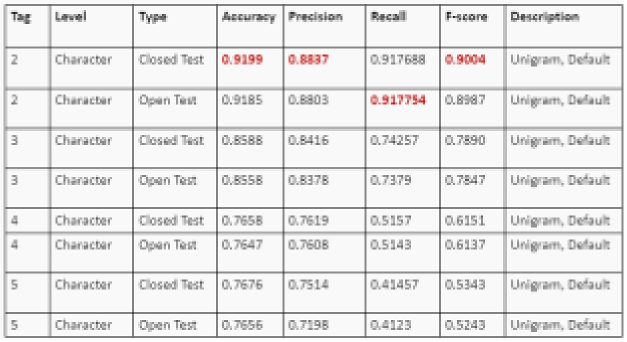
Recall

(sumi((TPi) / (TPi+FNi))) / number of unique elements in confusion matrix

F-score

(2*Recall*Precision)/(Recall+Precision)

Character-Level Segmentation Results

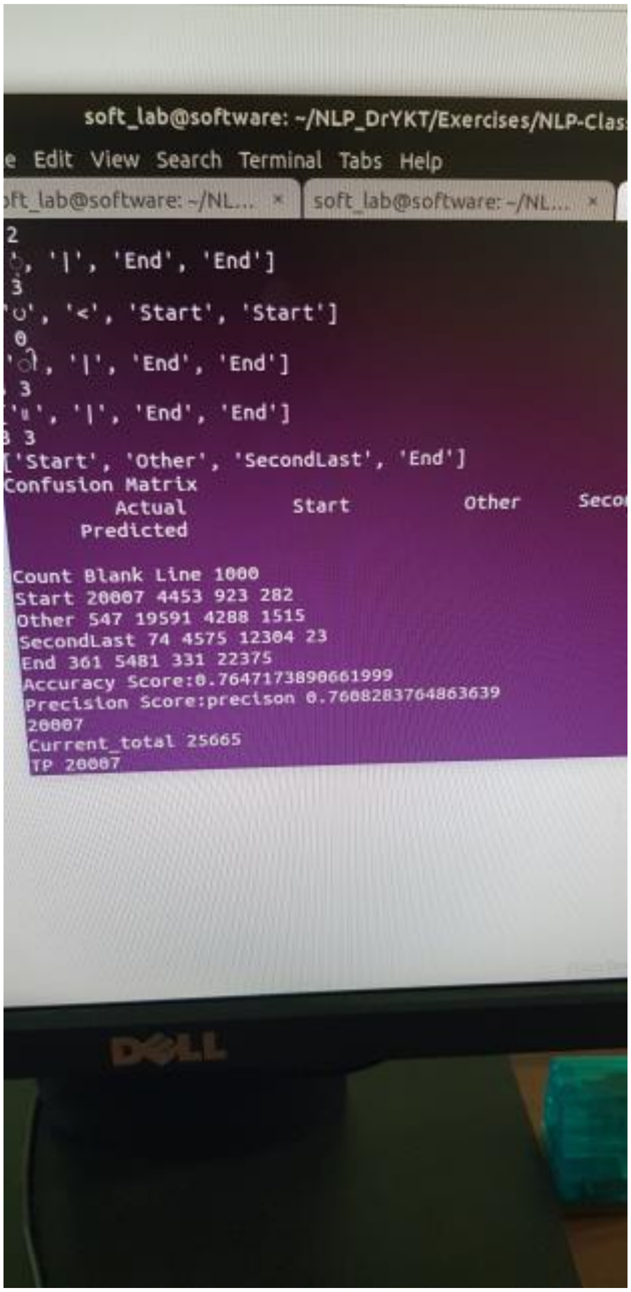


Tag	Level	Type	Accuracy	Precision	Recall	F-score	Description
2	Character	Closed Test	0.9199	0.8837	0.917688	0.9004	Unigram, Default
2	Character	Open Test	0.9185	0.8803	0.917754	0.8987	Unigram, Default
3	Character	Closed Test	0.8588	0.8416	0.74257	0.7890	Unigram, Default
3	Character	Open Test	0.8558	0.8378	0.7379	0.7847	Unigram, Default
4	Character	Closed Test	0.7658	0.7619	0.5157	0.6151	Unigram, Default
4	Character	Open Test	0.7647	0.7608	0.5143	0.6137	Unigram, Default
5	Character	Closed Test	0.7676	0.7514	0.41457	0.5343	Unigram, Default
5	Character	Open Test	0.7656	0.7198	0.4123	0.5243	Unigram, Default

Syllable-Level Segmentation Results

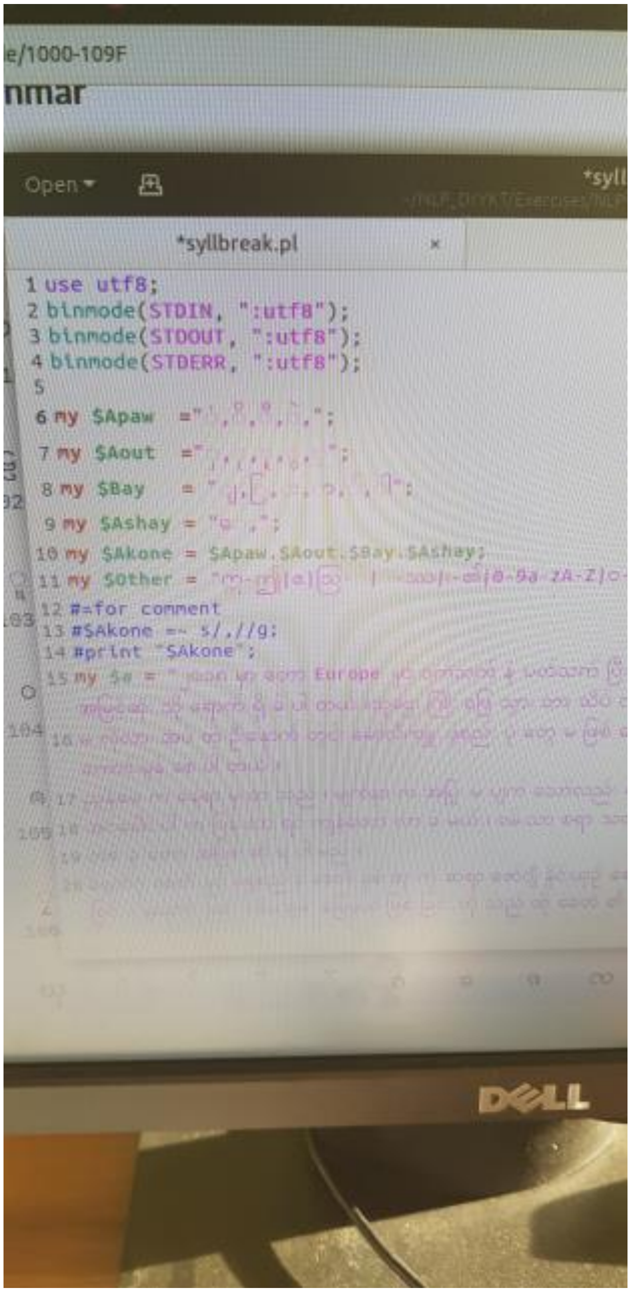
Soon....

Others


soft_lab@software: ~/NLP_DrYKT/Exercises/NLP-Clas
e Edit View Search Terminal Tabs Help
ft_lab@software: ~/NL...
soft_lab@software: ~/NL...
DELL

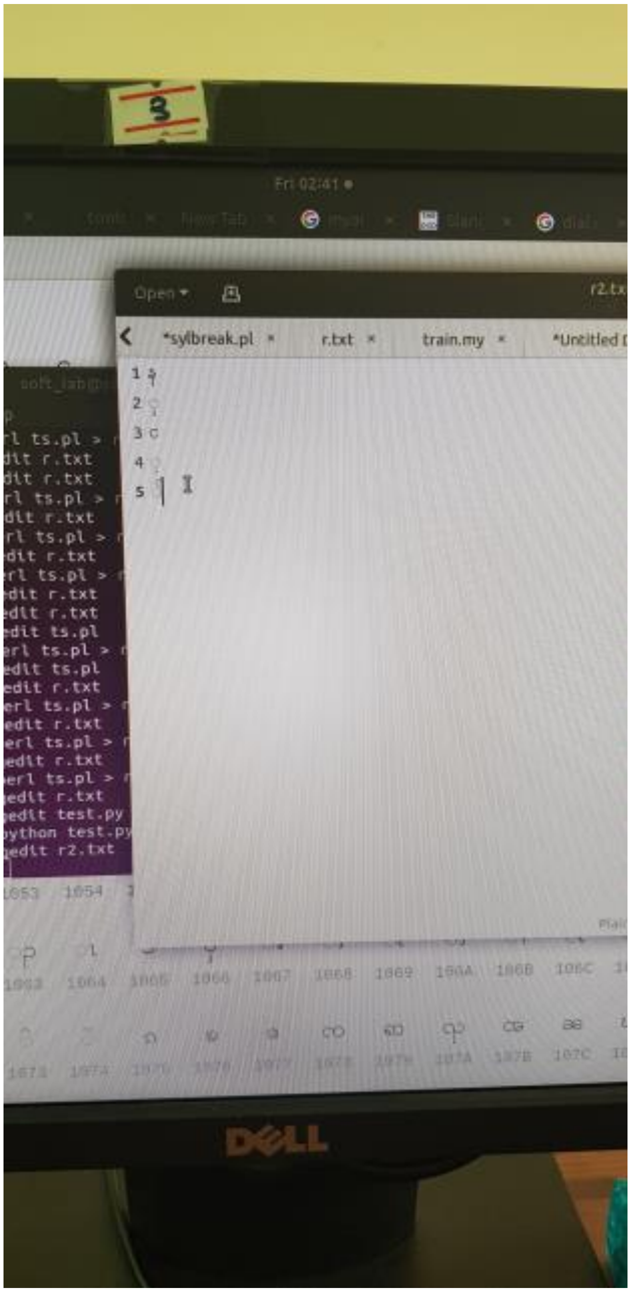
```
2
, '|', 'End', 'End']
 3
'', '<', 'Start', 'Start']
 0
'', '|', 'End', 'End']
 3
['', '|', 'End', 'End']
3 3
['Start', 'Other', 'SecondLast', 'End']
Confusion Matrix
         Actual          Start           Other       Seco
      Predicted

Count Blank Line 1000
Start 20007 4453 923 282
Other 547 19591 4288 1515
SecondLast 74 4575 12304 23
End 361 5481 331 22375
Accuracy Score:0.7647173890661999
Precision Score:precison 0.7608283764863639
20007
Current_total 25665
TP 20007
```

1 - EvaluationWithConfusionMatrix



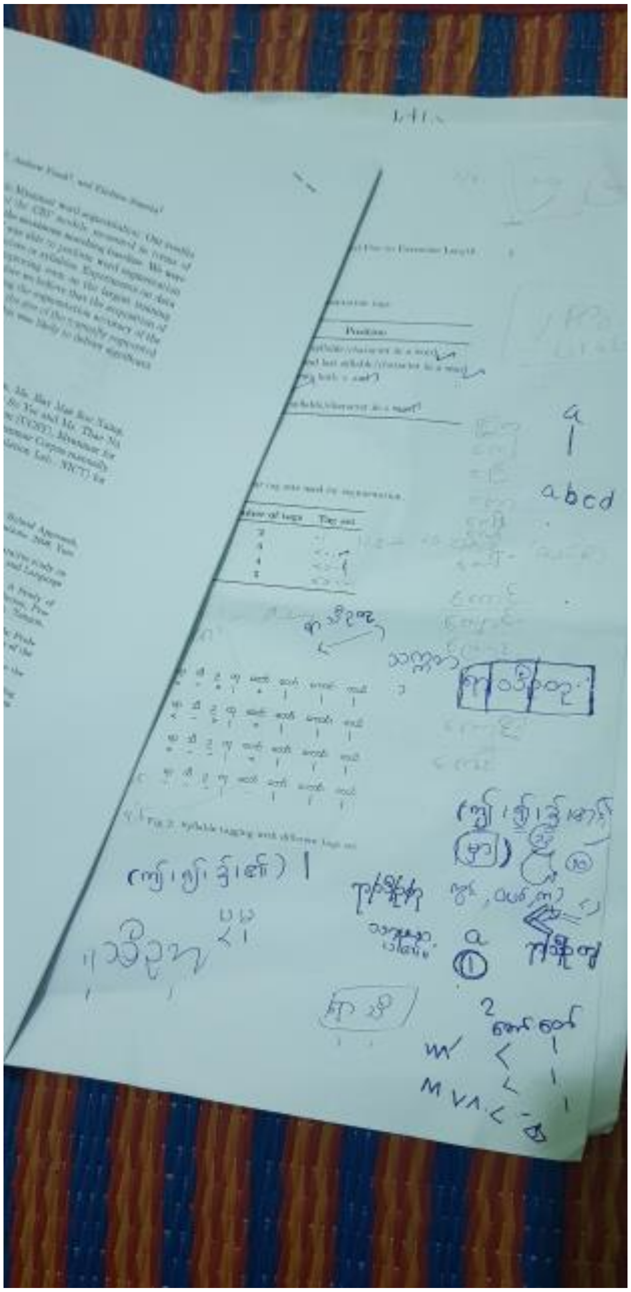
```
use utf8;
binmode(STDIN, ":utf8");
binmode(STDOUT, ":utf8");
binmode(STDERR, ":utf8");

my $Apaw  ="[illegible]";
my $Aout  ="[illegible]";
my $Bay   = "[illegible]";
my $Ashay = "[illegible]";
my $Akone = $Apaw.$Aout.$Bay.$Ashay;
my $Other = "[illegible]|[0-9a-zA-Z]
#=for comment
#$Akone =~ s/.//g;
#print "$Akone";
my $a = "[illegible] Europe [illegible]
```

2 - Thinking from different dimensions (syllbreak)



```
soft_lab@
p
rl ts.pl > r
dit r.txt
dit r.txt
rl ts.pl > r
dit r.txt
rl ts.pl > r
dit r.txt
erl ts.pl > r
edit r.txt
edit r.txt
edit ts.pl
erl ts.pl > r
edit ts.pl
edit r.txt
erl ts.pl > r
edit r.txt
erl ts.pl > r
edit r.txt
perl ts.pl > r
gedit r.txt
gedit test.py
python test.py
gedit r2.txt
```

3 - Checking order



4 - Paper

Thank You