WFST_Word_Segmentation_Small_Corpus

July 19, 2025

1 WFST Word Segmentation Tiny Demo

by Ye Kyaw Thu, Lab Leader, LST Lab., Myanmar
Date: 16 July 2025

ဒီ Notebook ရဲ့ အဓိက ရည်ရွယ်ချက်က WFST မော်ဒယ်တွေရဲ့အတွင်းပိုင်းကို မြင်သာအောင် visualization လုပ်ပြချင်တာပါ။
အဲဒါကြောင့် မော်ဒယ်တွေကို png, pdf ဖိုင်အနေနဲ့ ထုတ်နိုင်အော်လို့ ဒေတာကိုလည်း စာကြောင်း အနည်းငယ်ပဲ WFST word segmentation pipeline ကို အကြမ်းဆောက်သွားပါမယ်။

1.1 Reference

https://www.phontron.com/kyfd/tut1/

C++ နဲ့ ရေးထားတဲ့ kyfd က ကျောင်းသားတွေ installation လုပ်တဲ့အခါမှာ ခက်တာရယ် ပြီးတော့ perl code တွေထက် စာရင် လက်ရှိအချိန်မှာက Python programming နဲ့သွားတာက ပိုအဆင်ပြေတာမို့လို့ ဒီ Lab တစ်ခုလုံး run လို့ရအောင် fst_decoder.py နဲ့ preprocessing, post-task တွေအတွက် လိုအပ်တဲ့ code အားလုံးကို အချိန်ယူပြင်ဆင်ခဲ့တယ်။ သို့သော်လည်း code မှာ အမှားတချို့ ရှိနိုင်တာကို ဂရုပြုပါ။

1.2 Data Preparation

[2]:
```
cd /home/ye/exp/tiny_ws/
```

```
/home/ye/exp/tiny_ws
```

[4]:
```
!tree .
```

```
.
  fst_decoder.py
  script
      char_break.py
      evaluate.py
      evaluate_segmentation.py
      mk_lexicon.py
      mk_symbol.py
      rm_myanmar_punct.py
      word_vocab.py

2 directories, 8 files
```

[5]:
```
mkdir data
```

[6]:
```
cd data
```

```
/home/ye/exp/tiny_ws/data
```

1.2.1 Training Data

ဒီတခါတော့ training data ကို "မမ ဝဝ" တပုဒ်နဲ့ပဲ သွားကြရအောင်။

[9]:
```
!cat ./corpus.txt
```

```
မမ ဝဝ

မမ ဝဝ
ထထ က
အက ပထမ။

ကပါ ကပါ
မမ ရာ
ညည လသာသာ။

ည အခါ
ငါ စာ ရ
မမ ဝဝ
ထထ က။
```

Check filesize, content ...

[10]:
```
!wc corpus.txt
```

```
 15  23 184 corpus.txt
```

ဒီတခါတော့ **word segmentation** ကို **tool** တွေ ဘာတွေ မသုံးပဲ လက်နဲ့ပဲ ဖြတ်လိုက်တယ်။

[13]:
```
!cat corpus.ws
```

```
မမ ဝဝ
မမ ဝဝ
ထထ က
အက ပထမ ။
က ပါ က ပါ
မမ ရာ
ည ည လ သာ သာ ။
ည အခါ
ငါ စာ ရ
မမ ဝဝ
ထထ က ။
```

[14]:
```
!wc corpus.ws
```

```
 11  31 188 corpus.ws
```

1.3 Remove Myanmar Punctuation Characters

လက်ရှိ tiny corpus မှာလည်း ပုဒ်မက ပါနေသေးတာမို့လို့ data cleaning လုပ်ပါမယ်။

[16]:
```
cd ..
```

```
/home/ye/exp/tiny_ws
```

[19]:
```
!python ./script/rm_myanmar_punct.py < ./data/corpus.ws > ./data/corpus.clean
```

[20]:
```
!cat ./data/corpus.clean
```

```
မမ ဝဝ
မမ ဝဝ
ထထ က
အက ပထမ
က ပါ က ပါ
မမ ရာ
ည ည လ သာ သာ
ည အခါ
ငါ စာ ရ
မမ ဝဝ
ထထ က
```

1.3.1 Closed-Test Data

ဒီ ဒီမိုမှာတော့ closed test data ကိုလည်း သီးသန့်ပြင်မနေတော့ပဲ clean လုပ်ထားတဲ့ corpus ကိုပဲ ပြန်သုံးပြီး word segmentation မှန်မှန်ကန်ကန် လုပ်ပေးနိုင်သလားဆိုတာကိုပဲ confirmation လုပ်ပါမယ်။

1.4 Preparing a Lexicon

[23]:
```
!python ./script/word_vocab.py -help
```

```
usage: word_vocab.py [-h] [--input FILE] [--output FILE]

Build word vocabulary from input text

options:
  -h, --help     show this help message and exit
  --input FILE   Input text file (default: stdin)
  --output FILE  Output vocabulary file (default: stdout)
```

[40]:
```
!python ./script/word_vocab.py < ./data/corpus.clean > ./data/train.vocab
```

[41]:
```
!wc ./data/train.vocab
```

```
15 15 99 ./data/train.vocab
```

[42]:
```
!cat ./data/train.vocab
```

```
က
ငါ
စာ
ည
ထထ
ပထမ
ပါ
မမ
ရ
ရာ
လ
ဝဝ
သာ
အက
အခါ
```

1.5 Make a FST Format Lexicon

အထက်က ဆွဲထုတ်ခဲ့တဲ့ word တွေကို FST format အဖြစ် ပြောင်းမယ်။
အဲဒီအတွက် ရေးထားတဲ့ Python code ကိုလည်း လေ့လာကြည့်ပါ။

[29]:
```
!cat ./script/mk_lexicon.py
```

```
"""
Create lexicon FST from word list.
Written by Ye Kyaw thu, LU Lab., Myanmar.
Last updated: 11 July 2025
Usage:
    python mk_lexicon.py < train.vocab > lexicon.txt
"""

#!/usr/bin/env python3
import sys
import argparse
from collections import defaultdict

def main():
    parser = argparse.ArgumentParser(description='Create lexicon FST from word
list')
    parser.add_argument('--input', metavar='FILE',
                        help='Input word list file (default: stdin)',
                        type=argparse.FileType('r', encoding='utf-8'),
                        default=sys.stdin)
    parser.add_argument('--output', metavar='FILE',
```

```
                        help='Output FST file (default: stdout)',
                        type=argparse.FileType('w', encoding='utf-8'),
                        default=sys.stdout)

    args = parser.parse_args()

    newstate = 1

    try:
        for line in args.input:
            line = line.strip()
            if not line:
                continue

            # Prepare input symbols (characters + <w>)
            inputsym = list(line) + ['<w>']
            # Prepare output symbols (word + repeated <eps>)
            outputsym = [line] + ['<eps>'] * (len(inputsym) - 1)

            currstate = 0
            score = ' 1'

            while inputsym:
                nextstate = 0 if len(inputsym) == 1 else newstate
                if len(inputsym) > 1:
                    newstate += 1

                args.output.write(f"{currstate} {nextstate} {inputsym.pop(0)}
{outputsym.pop(0)}{score}\n")
                currstate = nextstate
                score = ''

        # Write the special transitions
        args.output.write("0 0 <s> <s>\n")
        args.output.write("0\n")

    except IOError as e:
        sys.stderr.write(f"Error: {e}\n")
        sys.exit(1)
    finally:
        if args.input is not sys.stdin:
            args.input.close()
        if args.output is not sys.stdout:
            args.output.close()

if __name__ == '__main__':
    main()
```

[43]:
```
!python ./script/mk_lexicon.py < ./data/train.vocab > ./data/lexicon.txt
```

[44]:
```
!cat ./data/lexicon.txt
```

```
0 1 က က 1
1 0 <w> <eps>
0 2 c ငါ 1
2 3 ါ <eps>
3 0 <w> <eps>
0 4 စ စာ 1
4 5 ာ <eps>
5 0 <w> <eps>
0 6 ည ည 1
6 0 <w> <eps>
0 7 ထ ထထ 1
7 8 ထ <eps>
8 0 <w> <eps>
0 9 ပ ပထမ 1
9 10 ထ <eps>
10 11 မ <eps>
11 0 <w> <eps>
0 12 ပ ပါ 1
12 13 ါ <eps>
13 0 <w> <eps>
0 14 မ မမ 1
14 15 မ <eps>
15 0 <w> <eps>
0 16 ရ ရ 1
16 0 <w> <eps>
0 17 ရ ရာ 1
17 18 ာ <eps>
18 0 <w> <eps>
0 19 လ လ 1
19 0 <w> <eps>
0 20 ဝ ဝဝ 1
20 21 ဝ <eps>
21 0 <w> <eps>
0 22 သ သာ 1
22 23 ာ <eps>
23 0 <w> <eps>
0 24 အ အက 1
24 25 က <eps>
25 0 <w> <eps>
0 26 အ အခါ 1
26 27 ခ <eps>
27 28 ါ <eps>
28 0 <w> <eps>
```

```
0 0 <s> <s>
0
```

1.6 Make Symbol Files

အရင်ဆုံး input လုပ်မယ့် ဖိုင်တွေနဲ့ ဆိုင်တဲ့ character level symbol ဖိုင်ကို ထုတ်မယ်။
input အတွက် ဆိုရင် –input ဆိုတဲ့ parameter ကို သုံးတယ်။

[33]:
```
!cat ./script/mk_symbol.py
```

```
"""
Generate input and output symbol tables.
Written by Ye Kyaw Thu, LU Lab., Myanmar.
Last update: 11 July 2025
Usage:
    python ../script/mk_symbol.py --input ./lexicon.txt > char.sym
    python ../script/mk_symbol.py --output ./lexicon.txt > word.sym
"""

#!/""usr/bin/""env python3
import sys
import argparse
from collections import defaultdict

def main():
    parser = argparse.ArgumentParser(description='Generate symbol table from FST
files')
    parser.add_argument('--input', metavar='FILE', action='append',
                        help='Input FST file(s) to read input symbols from')
    parser.add_argument('--output', metavar='FILE', action='append',
                        help='Output FST file(s) to read output symbols from')

    args = parser.parse_args()

    if not args.input and not args.output:
        parser.print_help()
        sys.exit(1)

    # Count the symbols to be printed
    toprint = defaultdict(int)

    special_symbols = ['<eps>', '<s>', '<unk>', '<w>']

    # Process input files
    if args.input:
        for fname in args.input:
            try:
                with open(fname, 'r', encoding='utf-8') as fstfile:
```

```
                    for line in fstfile:
                        line = line.strip()
                        arr = line.split()
                        if len(arr) >= 3:  # Ensure there's an input symbol
(index 2)
                            toprint[arr[2]] += 1
            except IOError as e:
                sys.stderr.write(f"Error opening file {fname}: {e}\n")
                sys.exit(1)

    # Process output files
    if args.output:
        for fname in args.output:
            try:
                with open(fname, 'r', encoding='utf-8') as fstfile:
                    for line in fstfile:
                        line = line.strip()
                        arr = line.split()
                        if len(arr) >= 4:  # Ensure there's an output symbol
(index 3)
                            toprint[arr[3]] += 1
            except IOError as e:
                sys.stderr.write(f"Error opening file {fname}: {e}\n")
                sys.exit(1)

    # Output symbols
    curr = 0
    for symbol in special_symbols:
        print(f"{symbol} {curr}")
        curr += 1
        toprint.pop(symbol, None)  # Remove if present

    # Output remaining symbols in sorted order
    for symbol in sorted(toprint.keys()):
        print(f"{symbol} {curr}")
        curr += 1

if __name__ == '__main__':
    main()
```

[45]:
```
!python ./script/mk_symbol.py --input ./data/lexicon.txt > ./data/char.sym
```

[46]:
```
!wc ./data/char.sym
```

```
 19  38 127 ./data/char.sym
```

Filesize က သေးတာမို့လို့ တစ်ဖိုင်လုံးပဲ ရိုက်ထုတ်ကြည့်ပြီး လေ့လာကြရအောင်။

[47]:
```
!cat ./data/char.sym
```

```
<eps> 0
<s> 1
<unk> 2
<w> 3
က 4
ခ 5
င 6
စ 7
ည 8
ထ 9
ပ 10
မ 11
ရ 12
လ 13
ဝ 14
သ 15
အ 16
ါ 17
ာ 18
```

ဒီတခါတော့ output symbol ဖိုင် သို့မဟုတ် word level unit symbol ဖိုင်ကို ထုတ်မယ်။
–output argument ကို သုံးရမယ်။

[48]:
```
!python ./script/mk_symbol.py --output ./data/lexicon.txt > ./data/word.sym
```

[49]:
```
!wc ./data/word.sym
```

```
 19  38 166 ./data/word.sym
```

[50]:
```
!cat ./data/word.sym
```

```
<eps> 0
<s> 1
<unk> 2
<w> 3
က 4
ငါ 5
စာ 6
ည 7
ထထ 8
ပထမ 9
ပါ 10
မမ 11
ရ 12
ရာ 13
လ 14
```

ဝဝ 15
သာ 16
အက 17
အခါ 18

1.7 Compilation

fstcompile ဆိုတဲ့ command ကိုသုံးပြီး compile လုပ်တာက text format ကနေ binary FST format အဖြစ် ပြောင်းတဲ့အလုပ်ပါ။

[52]:
```
!time fstcompile --isymbols=./data/char.sym --osymbols=./data/word.sym ./data/
 ↪lexicon.txt lexicon.fst
```

```
real    0m0.008s
user    0m0.006s
sys     0m0.002s
```

[53]:
```
!fstprint ./lexicon.fst
```

```
0       1       4       4       1
0       2       6       5       1
0       4       7       6       1
0       6       8       7       1
0       7       9       8       1
0       9       10      9       1
0       12      10      10      1
0       14      11      11      1
0       16      12      12      1
0       17      12      13      1
0       19      13      14      1
0       20      14      15      1
0       22      15      16      1
0       24      16      17      1
0       26      16      18      1
0       0       1       1
0
1       0       3       0
2       3       17      0
3       0       3       0
4       5       18      0
5       0       3       0
6       0       3       0
7       8       9       0
8       0       3       0
9       10      9       0
10      11      11      0
11      0       3       0
12      13      17      0
```

```
13      0       3       0
14      15      11      0
15      0       3       0
16      0       3       0
17      18      18      0
18      0       3       0
19      0       3       0
20      21      14      0
21      0       3       0
22      23      18      0
23      0       3       0
24      25      4       0
25      0       3       0
26      27      5       0
27      28      17      0
28      0       3       0
```

အထက်မှာ မြင်ရတဲ့အတိုင်း symbol file တွေကို မသတ်မှတ်ပေးတာကြောင့် ဂဏန်းနဲ့ပဲ ရိုက်ထုတ်ပြပါလိမ့်မယ်။ လူအနေနဲ့ စစ်ဆေးဖို့အတွက်က input symbol, output symbol ဖိုင်တွေကို သတ်မှတ်ပေးရပါလိမ့်မယ်။

[54]:
```
!fstprint --isymbols=./data/char.sym --osymbols=./data/word.sym ./lexicon.fst
```

```
0       1       က       က       1
0       2       င       ငါ      1
0       4       စ       စာ      1
0       6       ည       ည       1
0       7       ထ       ထထ      1
0       9       ပ       ပထမ     1
0       12      ပ       ပါ      1
0       14      မ       မမ      1
0       16      ရ       ရ       1
0       17      ရ       ရာ      1
0       19      လ       လ       1
0       20      ဝ       ဝဝ      1
0       22      သ       သာ      1
0       24      အ       အက      1
0       26      အ       အခါ     1
0       0       <s>     <s>
0
1       0       <w>     <eps>
2       3       ါ       <eps>
3       0       <w>     <eps>
4       5       ာ       <eps>
5       0       <w>     <eps>
6       0       <w>     <eps>
7       8       ထ       <eps>
8       0       <w>     <eps>
```

```
9       10      ∞       <eps>
10      11      ဓ       <eps>
11      0       <w>     <eps>
12      13      ◌ါ      <eps>
13      0       <w>     <eps>
14      15      ဓ       <eps>
15      0       <w>     <eps>
16      0       <w>     <eps>
17      18      ◌ာ      <eps>
18      0       <w>     <eps>
19      0       <w>     <eps>
20      21      ဝ       <eps>
21      0       <w>     <eps>
22      23      ◌ာ      <eps>
23      0       <w>     <eps>
24      25      ဘ       <eps>
25      0       <w>     <eps>
26      27      ခ       <eps>
27      28      ◌ါ      <eps>
28      0       <w>     <eps>
```

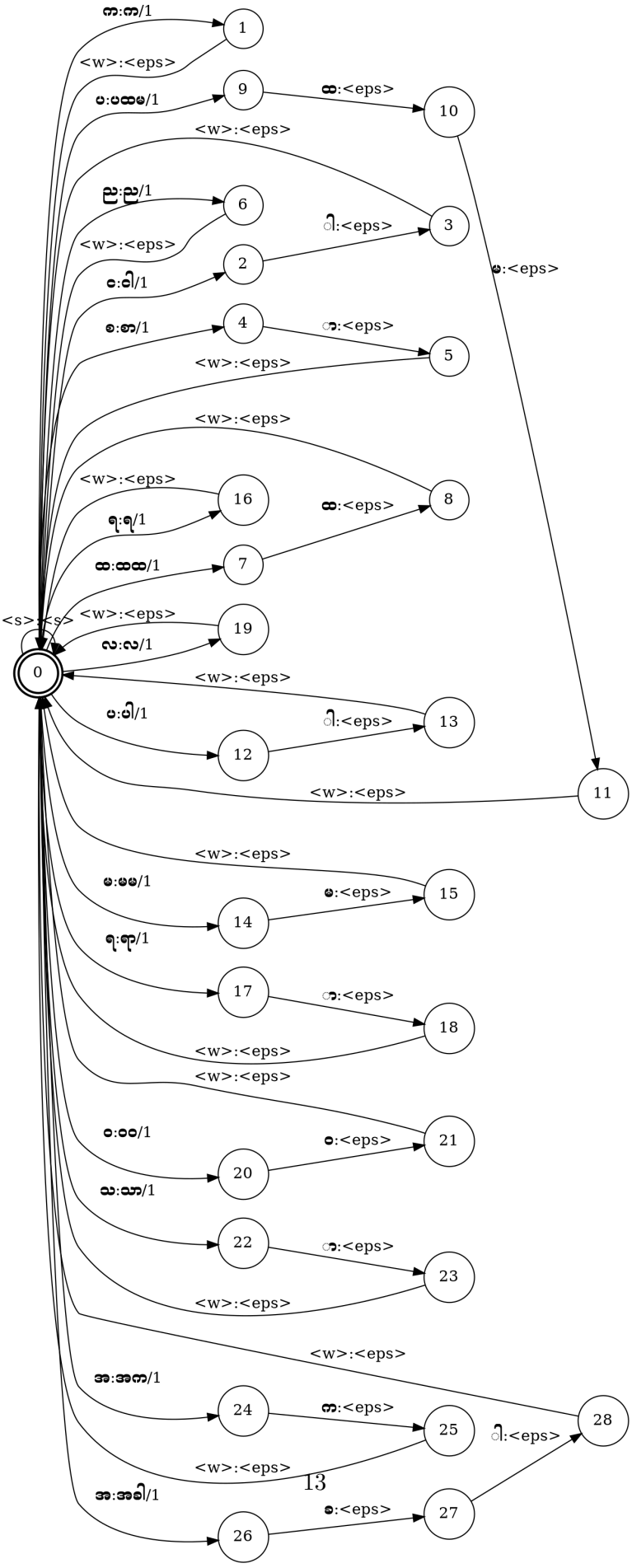
1.8 Visualization of Lexicon

[68]:
```
!fstdraw --portrait --isymbols=./data/char.sym --osymbols=./data/word.sym ./
 ↪lexicon.fst  | dot -Tpng -Gdpi=300 > lexicon.png
```

[72]:
```
!fstdraw --portrait --isymbols=./data/char.sym --osymbols=./data/word.sym ./
 ↪lexicon.fst  | dot -Tpdf > lexicon.pdf
```

[71]:
```
from IPython.display import Image
Image(filename="lexicon.png", width=800)  # Adjust width
```

[71]:


က:က/1
1
<w>:<eps>
ပ:ပဆဖ/1
9
ဆ:<eps>
10
<w>:<eps>
ည:ည/1
6
ါ:<eps>
3
<w>:<eps>
2
ဖ:<eps>
ဝ:ါ/1
စ:စ/1
4
ာ:<eps>
5
<w>:<eps>
<w>:<eps>
<w>:<eps>
16
8
ရ:ရ/1
ဆ:<eps>
ဆ:ဆဆ/1
7
<s>:<s>
<w>:<eps>
19
လ:လ/1
0
<w>:<eps>
ပ:ါ/1
ါ:<eps>
13
12
<w>:<eps>
11
<w>:<eps>
မ:ဓမ/1
15
ဓ:<eps>
14
ရ:ရော/1
17
ာ:<eps>
18
<w>:<eps>
<w>:<eps>
ဝ:ဝဝ/1
21
ဝ:<eps>
20
သ:သာ/1
22
ာ:<eps>
23
<w>:<eps>
<w>:<eps>
အ:အက/1
24
က:<eps>
25
ါ:<eps>
28
<w>:<eps>
အ:အါ/1
26
စ:<eps>
27

1.9 Optimization

ဒီနေရာမှာ OpenFST framework ရဲ့ အရေးကြီးတဲ့ command သုံးခုကိုသုံးပြီး model optimization လုပ်သွားပါမယ်။

1. determinize လုပ်တယ် ဆိုတာက non-deterministic FST တွေကို deterministic FST အဖြစ်ပြောင်းလဲတာပါ။ input နဲ့တွဲနေတဲ့ state တစ်ခုချင်းစီကို transition တစ်ခုဖြစ်အောင် ပြောင်းလဲတဲ့အပိုင်းပါ။ မဟုတ်ရင် input symbol တစ်ခုတည်းနဲ့ ချိတ်နေတဲ့ state တစ်ခုတည်းမှာ multiple transition တွေရှိနေတာမို့ network က ဖောင်းပွပြီး ဆုံးဖြတ်ချက်ချရတာမှာ ပိုခက်ခဲလို့ပါ။ Powerset construction algorithm ကိုသုံးပါတယ်။

2. minimize လုပ်တယ် ဆိုတာက မော်ဒယ်တစ်ခုလုံးမှာရှိတဲ့ state အရေအတွက်၊ arc အရေအတွက်တွေကို လျှော့ချတဲ့အပိုင်းပါ။ Hopcroft's algorithm ကိုသုံးပြီး အလုပ်လုပ်ပုံ တူညီတဲ့ state တွေကို merge လုပ်ပါတယ်။
3. arc တွေကို sorting လုပ်တဲ့ အပိုင်းပါ။

[73]:
```
!fstdeterminize lexicon.fst lexicon.det
```

[74]:
```
!fstminimize lexicon.det lexicon.min
```

[75]:
```
!fstarcsort lexicon.min lexicon.srt
```

Optimization မလုပ်ခင်က .fst ဖိုင်နဲ့ .srt ဖိုင်ရဲ့ filesize ကို နှိုင်းယှဉ်ကြည့်ရင် ကွာတာကို မြင်ရပါလိမ့်မယ်။

[76]:
```
!ls -lh lexicon.fst lexicon.srt
```

```
-rw-rw-r-- 1 ye ye 1.1K Jul 16 21:19 lexicon.fst
-rw-rw-r-- 1 ye ye  586 Jul 16 22:18 lexicon.srt
```

srt ဖိုင်ကို text ဖိုင်အဖြစ်ပြောင်းလို့ ရတယ်။

[77]:
```
!fstprint --isymbols=./data/char.sym --osymbols=./data/word.sym lexicon.srt >␣
↪lexicon.srt.txt
```

[78]:
```
!cat lexicon.srt.txt
```

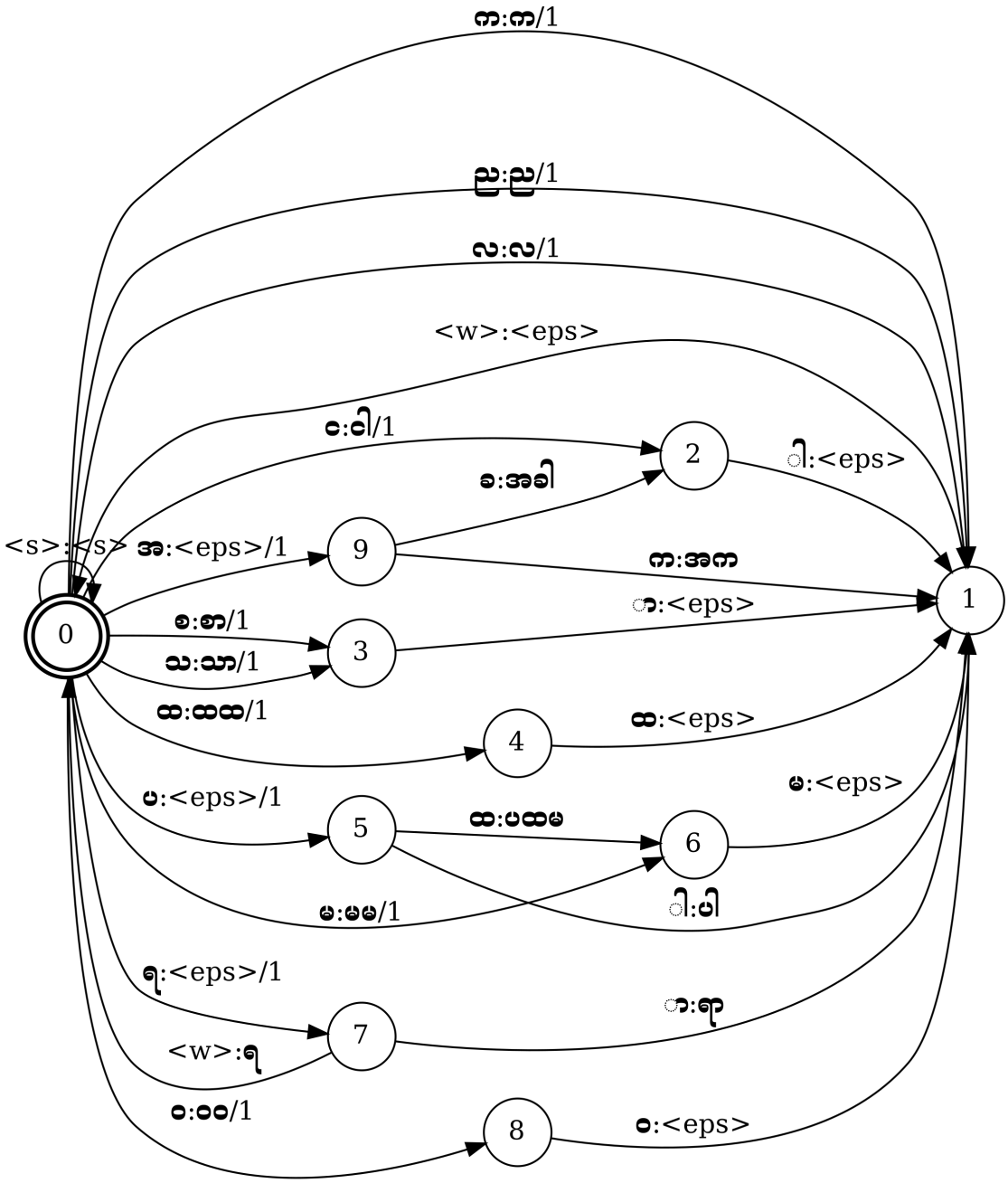
```
0       0       <s>     <s>
0       1       က       က       1
0       2       c       ငါ      1
0       3       စ       စာ      1
0       1       ည       ည       1
0       4       ထ       ထထ      1
0       5       ပ       <eps>   1
0       6       မ       မမ      1
0       7       ရ       <eps>   1
0       1       လ       လ       1
```

```
0       8       ဝ       ဝဝ      1
0       3       သ       သာ      1
0       9       အ       <eps>   1
0
1       0       <w>     <eps>
2       1       ါ       <eps>
3       1       ာ       <eps>
4       1       တ       <eps>
5       6       တ       ပထမ
5       1       ါ       ပါ
6       1       မ       <eps>
7       0       <w>     ရ
7       1       ာ       ရာ
8       1       ဝ       <eps>
9       1       က       အက
9       2       ခ       အခါ
```

[79]:
```
!fstdraw --portrait --isymbols=./data/char.sym --osymbols=./data/word.sym ./
 ↪lexicon.srt  | dot -Tpng -Gdpi=300 > lexicon.srt.png
```

[80]:
```
from IPython.display import Image
Image(filename="lexicon.srt.png", width=800)  # Adjust width
```

[80]:



1.10 Preparation for Decoding

ဒီနေရာမှာ word boundary ကိစ္စကို ဘယ်လိုသတ်မှတ်ကြမလဲ စဉ်းစားရလိမ့်မယ်။
ဆိုတဲ့ သင်္ကေတကိုပဲသုံးပြီး စာလုံးတိုင်းရဲ့ နောက်မှာ လိုက်ဖြည့်ရင်လည်း ရပေမဲ့ lexicon or dictionry ဖိုင်ကလည်း ရှုပ်ထွေးမှုကို နဲ့ အစားထိုးလုပ်သွားမယ်။

[81]:
```
!echo "3 0" > map.txt
```

[82]:
```
!cat map.txt
```

```
3 0
```

[83]:
```
!fstrelabel --relabel_ipairs=map.txt lexicon.srt lexicon_final.fst
```

ပြန်စစ်ကြည့်ရအောင်

[84]:
```
!fstprint --isymbols=./data/char.sym --osymbols=./data/word.sym lexicon_final.
↪fst > lexicon_final.fst.txt
```

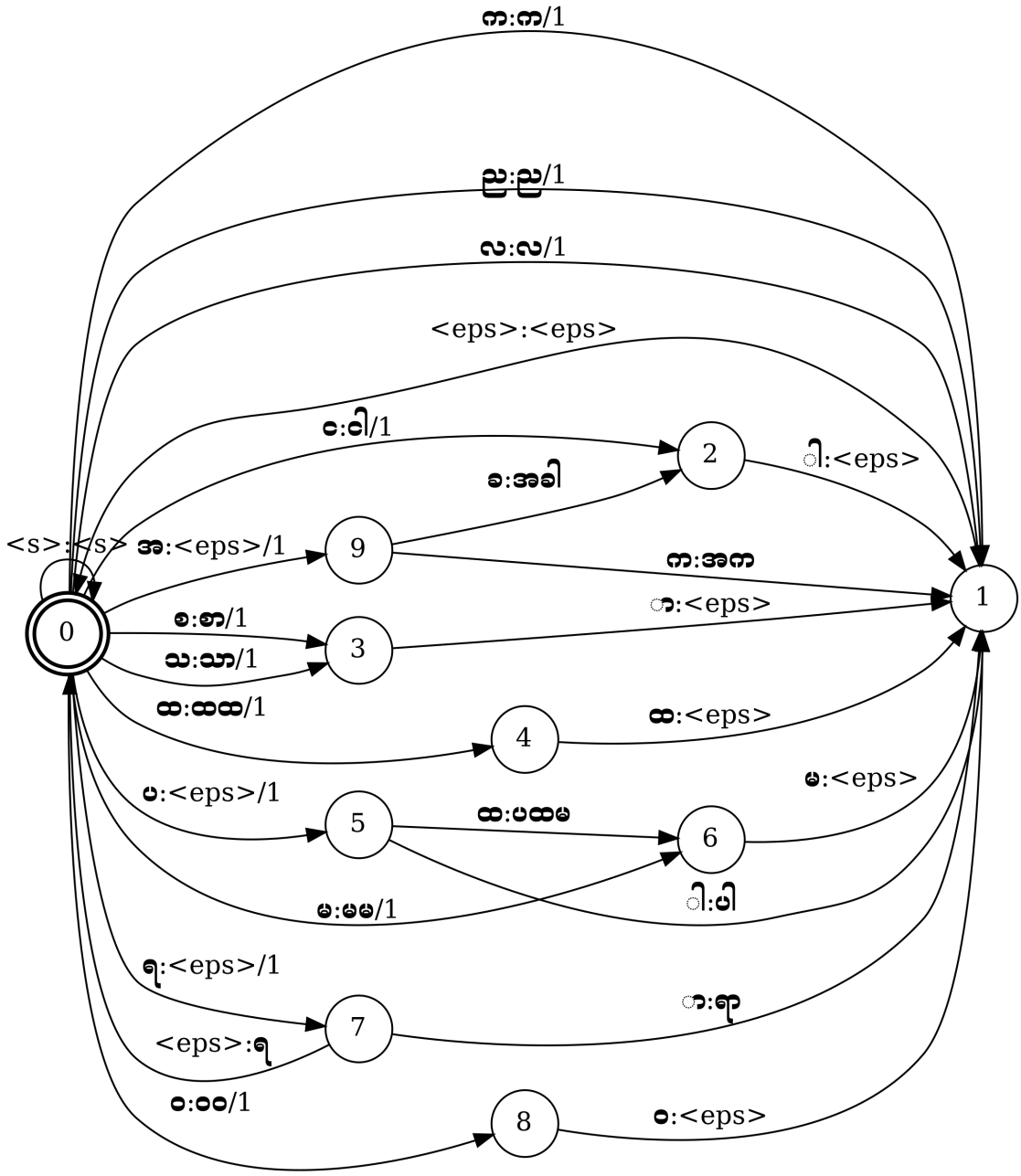
[85]:
```
!cat ./lexicon_final.fst.txt
```

```
0       0       <s>     <s>
0       1       က       က       1
0       2       c       ငါ      1
0       3       စ       စာ      1
0       1       ည       ည       1
0       4       ထ       ထထ      1
0       5       ပ       <eps>   1
0       6       မ       မမ      1
0       7       ရ       <eps>   1
0       1       လ       လ       1
0       8       ဝ       ဝဝ      1
0       3       သ       သာ      1
0       9       အ       <eps>   1
0
1       0       <eps>   <eps>
2       1       ါ       <eps>
3       1       ာ       <eps>
4       1       ထ       <eps>
5       6       ထ       ပထမ
5       1       ါ       ပါ
6       1       မ       <eps>
7       0       <eps>   ရ
7       1       ာ       ရာ
8       1       ဝ       <eps>
9       1       က       အက
9       2       ခ       အခါ
```

[87]:
```
!fstdraw --portrait --isymbols=./data/char.sym --osymbols=./data/word.sym ./
↪lexicon_final.fst  | dot -Tpng -Gdpi=300 > lexicon_final.fst.png
```

[88]:
```
from IPython.display import Image
Image(filename="lexicon_final.fst.png", width=800)  # Adjust width
```

[88]:



1.11 Configuration File

fst_decoder.py ကို မသုံးခင်မှာ configuration ဖိုင်ကို ကြိုတင်ပြင်ဆင်ထားဖို့ လိုအပ်တယ်။ yaml format ကို သုံးထားတယ်။

[92]:
```
!cat config.yaml
```

```
input_format: text
output_format: text
```

```
nbest: 1
input_symbols: ./data/char.sym
output_symbols: ./data/word.sym
unknown_symbol: <unk>
terminal_symbol: </s>
models:
  - type: plain
    file: ./lexicon_final.fst
    input_symbols: ./data/char.sym
    output_symbols: ./data/word.sym
    weights: [1.0]
```

1.12 Character Segmentation

Input လုပ်တဲ့အခါမှာ character တစ်လုံးချင်းစီဖြတ်ထားတဲ့ ပုံစံနဲ့ input လုပ်မှာမို့ test data ဖိုင်တွေကို character segmentation လုပ်ဖို့ လိုအပ်တယ်။

[93]:
```
!python ./script/char_break.py < ./data/corpus.clean > ./data/corpus.char
```

[94]:
```
!cat ./data/corpus.char
```

```
မ မ ဝ ဝ
မ မ ဝ ဝ
ထ ထ က
အ က ပ ထ မ
က ပ ါ က ပ ါ
မ မ ရ ာ
ည ည လ သ ာ သ ာ
ည အ ခ ါ
ငါ စ ာ ရ
မ မ ဝ ဝ
ထ ထ က
```

1.13 Decoding

လက်ရှိ လုပ်ပြတဲ့ ဒီမိုမှာတော့ decoding လုပ်တယ်ဆိုတာက word segmentation လုပ်တာပေါ့။ fst_decoder.py code ကို လေ့လာချင်သူတွေအတွက် cat နဲ့ ရိုက်ထုတ်ပြမယ်။

[95]:
```
!cat fst_decoder.py
```

```
"""
FST Decoder
Written by Ye Kyaw Thu
Last Update: 11 July 2025

Usage:
    python ./fst_decoder.py ./config.yaml --show-id < ./closed_test.txt >
closed_test.hyp
```

```
    python ./fst_decoder.py ./config.yaml --show-id < ./open_test.txt >
open_test.hyp

Reference: https://www.phontron.com/kyfd/
"""

import os
import sys
import yaml
import argparse
import time
from typing import List, Dict, Any
import pywrapfst as fst
import pynini
import tempfile

class ConfigError(Exception):
    pass

class DecoderError(Exception):
    pass

class FSTModel:
    def __init__(self, config, symbol_tables=None):
        """Initialize the FST model with configuration"""
        self.config = config
        self.symbol_tables = symbol_tables or {}
        self.fst = None

    def load(self):
        """Load the FST with explicit symbol tables"""
        try:
            fst_file = self.config.get('file')
            if not fst_file:
                raise ConfigError("No FST file specified in config")

            if not os.path.exists(fst_file):
                raise ConfigError(f"FST file not found: {fst_file}")

            print(f"Attempting to load FST from: {fst_file}", file=sys.stderr)

            # Load FST first
            try:
                self.fst = pynini.Fst.read(fst_file)
            except Exception as e:
                raise ConfigError(f"Failed to read FST file {fst_file}:
{str(e)}")
```

```
            if self.fst.start() == -1:
                raise ConfigError("Loaded FST has no start state")

            # Attach symbol tables if paths are provided
            input_sym_path = self.config.get('input_symbols')
            if input_sym_path:
                if not os.path.exists(input_sym_path):
                    print(f"Warning: Input symbol table not found:
{input_sym_path}", file=sys.stderr)
                else:
                    input_sym = pynini.SymbolTable.read_text(input_sym_path)
                    self.fst.set_input_symbols(input_sym)

            output_sym_path = self.config.get('output_symbols')
            if output_sym_path:
                if not os.path.exists(output_sym_path):
                    print(f"Warning: Output symbol table not found:
{output_sym_path}", file=sys.stderr)
                else:
                    output_sym = pynini.SymbolTable.read_text(output_sym_path)
                    self.fst.set_output_symbols(output_sym)

            print(f"Successfully loaded FST with {self.fst.num_states()}
states", file=sys.stderr)
            print(f"Input symbols: {'yes' if self.fst.input_symbols() else
'no'}", file=sys.stderr)
            print(f"Output symbols: {'yes' if self.fst.output_symbols() else
'no'}", file=sys.stderr)

        except Exception as e:
            raise ConfigError(f"Error loading FST: {str(e)}")

    def verify(self):
        """Verify the loaded FST meets basic requirements"""
        if self.fst is None:
            raise ConfigError("FST not loaded")
        if self.fst.start() == -1:
            raise ConfigError("FST has no start state")
        if self.fst.num_states() == 0:
            raise ConfigError("FST has no states")
        return True

class DecoderConfig:
    def __init__(self, config_file: str = None, args: Dict = None):
        self.config = {
            'input_format': 'text',
            'output_format': 'text',
            'nbest': 1,
```

```
            'beam_width': 0,
            'trim_width': 0.0,
            'print_duplicates': False,
            'print_input': False,
            'print_all': False,
            'sample': False,
            'negative_probs': False,
            'weights': [1.0],
            'input_symbols': None,
            'output_symbols': None,
            'unknown_symbol': '<unk>',
            'terminal_symbol': '</s>',
            'show_id': False,
            'models': []
        }

        self.symbol_tables = {'input': None, 'output': None}
        self.unknown_id = -1
        self.terminal_id = -1

        if config_file:
            self.load(config_file)
        if args:
            self._apply_args(args)  # Changed from apply_args to _apply_args

    def load(self, config_file: str):
        try:
            with open(config_file, 'r') as f:
                config_data = yaml.safe_load(f) or {}

            for key, value in config_data.items():
                if key in self.config:
                    self.config[key] = value

            # Load symbol tables with verification
            if self.config['input_symbols']:
                self.symbol_tables['input'] =
fst.SymbolTable.read_text(self.config['input_symbols'])
                if self.config['unknown_symbol']:
                    self.unknown_id =
self.symbol_tables['input'].find(self.config['unknown_symbol'])
                if self.config['terminal_symbol']:
                    self.terminal_id =
self.symbol_tables['input'].find(self.config['terminal_symbol'])

            if self.config['output_symbols']:
                self.symbol_tables['output'] =
fst.SymbolTable.read_text(self.config['output_symbols'])
```

```
        except Exception as e:
            raise ConfigError(f"Config loading error: {str(e)}")

    def _apply_args(self, args: Dict):
        """Apply command-line arguments to configuration"""
        for key, value in args.items():
            if value is not None and key in self.config:
                # Handle special cases
                if key == 'weights':
                    self.config[key] = [float(w) for w in value.split(',')]
                elif key in ['nbest', 'beam_width']:
                    self.config[key] = int(value)
                elif key == 'trim_width':
                    self.config[key] = float(value)
                elif key in ['print_input', 'print_all', 'sample',
'negative_probs', 'print_duplicates']:
                    self.config[key] = bool(value)
                elif key == 'show_id':
                    self.config[key] = bool(value)
                else:
                    self.config[key] = value

class Decoder:
    def __init__(self, config: DecoderConfig):
        self.config = config
        self.sentence_id = 0
        self.multiplier = -1 if config.config['negative_probs'] else 1
        self.unknown_words = []

        # Initialize models with verification
        self.models = []

        for model_config in self.config.config['models']:
            try:
                print(f"\nInitializing model with config: {model_config}",
file=sys.stderr)
                model = FSTModel(model_config, self.config.symbol_tables)
                print("Model object created, attempting to load…",
file=sys.stderr)
                model.load()
                # In Decoder.__init__, after model.load():
                model.verify()
                if model.fst is None:
                    raise DecoderError(f"Model loaded but fst is None:
{model_config.get('file', 'unknown')}")

                # Additional verification
```

```
                print(f"Model loaded successfully. Start state:
{model.fst.start()}", file=sys.stderr)
                print(f"Num states: {model.fst.num_states()}", file=sys.stderr)
                print(f"Input symbols: {model.fst.input_symbols()}",
file=sys.stderr)
                print(f"Output symbols: {model.fst.output_symbols()}",
file=sys.stderr)

                self.models.append(model)
            except Exception as e:
                raise DecoderError(f"Model initialization failed: {str(e)}")

    def decode(self, input_stream, output_stream) -> bool:
        try:
            print("-- Starting FST Decoder --", file=sys.stderr)

            if self.config.config['input_format'] == 'text':
                return self._process_text(input_stream, output_stream)
            else:
                raise DecoderError("FST input format not yet implemented")

        except Exception as e:
            print(f"ERROR: {str(e)}", file=sys.stderr)
            return False

    def _process_text(self, input_stream, output_stream) -> bool:

        for lineno, line in enumerate(input_stream):
            tokens = line.strip().split()
            if not tokens:
                continue

            try:
                input_fst = self._make_input_fst(tokens)
                best_fst = self._find_best_paths(input_fst)

                if best_fst.start() == -1:
                    print("WARNING: no path found", file=sys.stderr)
                    continue

                self._print_result(best_fst, output_stream, lineno)
                self.sentence_id += 1

            except Exception as e:
                #print(f"ERROR processing sentence: {str(e)}", file=sys.stderr)
                if self.config.config['show_id']:
                    print(f"# Skipped line {lineno}: {tokens}", file=sys.stderr)
```

```
            continue

        return True

    def _make_input_fst(self, tokens: List[str]):
        """Create input FST using symbol table mapping"""
        try:
            # Create empty FST with symbol tables
            input_fst = pynini.Fst()
            input_sym = self.config.symbol_tables['input']
            input_fst.set_input_symbols(input_sym)
            input_fst.set_output_symbols(input_sym)

            start_state = input_fst.add_state()
            input_fst.set_start(start_state)
            current_state = start_state
            self.unknown_words = []

            for token in tokens:
                next_state = input_fst.add_state()
                label = input_sym.find(token)

                if label == -1:
                    if self.config.unknown_id == -1:
                        raise DecoderError(f"Unknown token '{token}'")
                    label = self.config.unknown_id
                    self.unknown_words.append(token)

                input_fst.add_arc(current_state,
                                  pynini.Arc(label, label, 0.0, next_state))
                current_state = next_state

            input_fst.set_final(current_state, 0.0)
            return input_fst

        except Exception as e:
            raise DecoderError(f"Input FST creation failed: {str(e)}")

    def _find_best_paths(self, input_fst):
        """Find best paths with symbol verification"""
        try:
            # Verify symbol tables
            if not input_fst.input_symbols():
                raise DecoderError("Input FST missing input symbols")
            if not self.models[0].fst.input_symbols():
                raise DecoderError("Model FST missing input symbols")

            # Check symbol table compatibility
```

```
            input_symtab = input_fst.input_symbols()
            model_symtab = self.models[0].fst.input_symbols()

            input_syms = set(input_symtab.find(i) for i in
range(input_symtab.num_symbols()))
            model_syms = set(model_symtab.find(i) for i in
range(model_symtab.num_symbols()))


            common_syms = input_syms & model_syms
            print(f"Common symbols: {len(common_syms)}", file=sys.stderr)

            search_fst = input_fst.copy()

            for model in self.models:
                print(f"Composing with model: {model.config['file']}",
file=sys.stderr)

                # Perform composition
                composed = pynini.compose(search_fst, model.fst)
                if composed.start() == -1:
                    print("Composition failed - possible symbol mismatch",
file=sys.stderr)
                    print(f"Input FST symbols: {input_fst.input_symbols()}",
file=sys.stderr)
                    print(f"Model FST symbols: {model.fst.input_symbols()}",
file=sys.stderr)
                    return composed

                search_fst = composed

            return pynini.shortestpath(search_fst)

        except Exception as e:
            raise DecoderError(f"Path finding error: {str(e)}")

    def _print_result(self, result_fst, output_stream, lineno: int):
        """Print results with proper symbol table handling"""
        try:
            if result_fst.start() == -1:
                print("WARNING: no path found", file=sys.stderr)
                return

            # Convert to string using symbol tables
            output_str = ""
            state = result_fst.start()
            while state != -1:
                arcs = list(result_fst.arcs(state))
```

```
                if not arcs:
                    break

                arc = arcs[0]  # Take first arc
                olabel = arc.olabel

                # Handle output symbols
                if self.config.symbol_tables['output']:
                    out_sym = self.config.symbol_tables['output'].find(olabel)

                    if out_sym == "<eps>" or out_sym == -1:
                        pass  # skip epsilon and undefined
                    else:
                        output_str += f"{out_sym} "
                else:
                    if olabel != 0:
                        output_str += f"{olabel} "


                state = arc.nextstate

            # Write output
            #line = f"{self.sentence_id}|||{output_str.strip()}"
            #print(line, file=output_stream)
            #print(output_str.strip(), file=output_stream)
            if self.config.config['show_id']:
                print(f"{lineno}|||{output_str.strip()}", file=output_stream)
            else:
                print(output_str.strip(), file=output_stream)

        except Exception as e:
            raise DecoderError(f"Output generation error: {str(e)}")

    def _get_input_symbol(self, label: int, unk_idx: int) -> str:
        if label == self.config.unknown_id:
            if unk_idx < len(self.unknown_words):
                return self.unknown_words[unk_idx]
            return "<unk>"
        return self.config.symbol_tables['input'].find(label) or str(label)

    def _get_output_symbol(self, label: int, unk_idx: int) -> str:
        if label == self.config.unknown_id:
            if unk_idx < len(self.unknown_words):
                return self.unknown_words[unk_idx]
            return "<unk>"
        return self.config.symbol_tables['output'].find(label) or str(label)

def parse_args():
```

```
    parser = argparse.ArgumentParser(description="KYFD - A WFST-based decoder")
    parser.add_argument("config", help="Configuration file (YAML format)")
    parser.add_argument("-i", "--input", choices=["text", "fst"],
dest="input_format",
                        help="Input format (text or fst)")
    parser.add_argument("-o", "--output", choices=["text", "score",
"component"], dest="output_format",
                        help="Output format")
    parser.add_argument("-n", "--nbest", type=int, help="Number of best paths to
output")
    parser.add_argument("-w", "--weights", help="Comma-separated list of
weights")
    parser.add_argument("-u", "--unknown", help="Unknown symbol")
    parser.add_argument("-t", "--terminal", help="Terminal symbol")
    parser.add_argument("--beam", type=int, dest="beam_width", help="Beam
width")
    parser.add_argument("--trim", type=float, dest="trim_width", help="Trim
threshold")
    parser.add_argument("--print-input", action="store_true", help="Print input
sequence")
    parser.add_argument("--print-all", action="store_true", help="Print all
arcs")
    parser.add_argument("--sample", action="store_true", help="Sample paths
instead of shortest path")
    parser.add_argument("--negative", action="store_true",
dest="negative_probs",
                        help="Treat weights as negative log probabilities")
    parser.add_argument("--show-id", action="store_true", help="Show original
line number in output")


    return parser.parse_args()

def main():
    args = parse_args()

    try:
        config = DecoderConfig(args.config, vars(args))
        decoder = Decoder(config)
        success = decoder.decode(sys.stdin, sys.stdout)
        sys.exit(0 if success else 1)
    except Exception as e:
        print(f"FATAL ERROR: {str(e)}", file=sys.stderr)
        sys.exit(1)

if __name__ == "__main__":
    main()
```

1.14 Decoding with Closed Test Data

[96]:
```
!time python ./fst_decoder.py ./config.yaml < ./data/corpus.char > ./closed.hyp
```

```
Initializing model with config: {'type': 'plain', 'file': './lexicon_final.fst',
'input_symbols': './data/char.sym', 'output_symbols': './data/word.sym',
'weights': [1.0]}
Model object created, attempting to load…
Attempting to load FST from: ./lexicon_final.fst
Successfully loaded FST with 10 states
Input symbols: yes
Output symbols: yes
Model loaded successfully. Start state: 0
Num states: 10
Input symbols: <const Fst SymbolTableView './data/char.sym' at 0x785b4a3339c0>
Output symbols: <const Fst SymbolTableView './data/word.sym' at 0x785b4a3339c0>
-- Starting FST Decoder --
Common symbols: 19
Composing with model: ./lexicon_final.fst
Common symbols: 19
Composing with model: ./lexicon_final.fst
Common symbols: 19
Composing with model: ./lexicon_final.fst
Common symbols: 19
Composing with model: ./lexicon_final.fst
Common symbols: 19
Composing with model: ./lexicon_final.fst
Common symbols: 19
Composing with model: ./lexicon_final.fst
Common symbols: 19
Composing with model: ./lexicon_final.fst
Common symbols: 19
Composing with model: ./lexicon_final.fst
Common symbols: 19
Composing with model: ./lexicon_final.fst
Common symbols: 19
Composing with model: ./lexicon_final.fst
Common symbols: 19
Composing with model: ./lexicon_final.fst
Common symbols: 19
Composing with model: ./lexicon_final.fst

real    0m0.095s
user    0m0.051s
sys     0m0.014s
```

[97]:
```
!wc closed.hyp
```

```
 11  28 175 closed.hyp
```

[98]:
```
!cat closed.hyp
```

```
မမ ဝဝ
မမ ဝဝ
ထထ က
အက ပထမ
က ပါ က ပါ
မမ ရာ
ည ည လ သာ သာ
ည အခါ
ငါ စာ ရ
မမ ဝဝ
ထထ က
```

1.15 Evaluation for Closed-Test Data

ဒီ evaluation script က ဆရာ NICT, Kyoto က Multilingual Machine Translation Lab. မှာ အလုပ်လုပ်ကတည်းက သုံးခဲ့တဲ့ python code ပါ။ အဲဒါကြောင့် run မယ် ဆိုရင် python2.7 နဲ့မှ အဆင်ပြေပါတယ်။ ရေးထားတဲ့ Author က Yue Zhang, University of Oxford ပါ။
code ကို လေ့လာချင်သူတွေအတွက် cat နဲ့ ရိုက်ထုတ်ပြမယ်။

[99]:
```
!cat ./script/evaluate.py
```

```
#****************************************************************
#
# evaluate.py - the evaluation program.
#
# Author: Yue Zhang
#
# Computing lab, University of Oxford. 2006.11
#
#****************************************************************

#===============================================================
#
# Import modules.
#
#===============================================================

import sys
import os
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), "..",
"libs")))
import getopt

#---------------------------------------------------------------
#
```

```
# addTuples - add two tuples element by element.
#
# Inputs:  tuple1 - operand1
#          tuple2 - operand2
#
# Returns: tuple
#
#-----------------------------------------------------------------

def addTuples(tuple1, tuple2):
   return tuple([tuple1[i]+tuple2[i] for i in xrange(len(tuple1))])

#-----------------------------------------------------------------
#
# addListToList - add the second list to the first list.
#
# Inputs:  list1 - operand1, the one modified list
#          list2 - operand2, the list added
#
#-----------------------------------------------------------------

def addListToList(list1, list2):
   for i in xrange(len(list1)):
      list1[i] += list2[i]

#-----------------------------------------------------------------
#
# subtractListFromList - subtract the second list from the first list.
#
# Inputs:  list1 - operand1, the one modified list
#          list2 - operand2, the list added
#
#-----------------------------------------------------------------

def subtractListFromList(list1, list2):
   for i in xrange(len(list1)):
      list1[i] -= list2[i]

#-----------------------------------------------------------------
#
# dotProduct - compute the dot-product for lists/tuples.
#
# Inputs:  list1 - operand1
#          list2 - operand2
#
# Returns: int
#
#-----------------------------------------------------------------
```

```
def dotProduct(list1, list2):
   nReturn = 0
   for i in xrange(len(list1)):
      nReturn += list1[i] * list2[i]
   return nReturn

#-----------------------------------------------------------------
#
# addDictToDict - add a dictionary with int value to another dictionary
#
# Input: dict1 - operand1, the one that is added to
#        dict2 - operand2, the one to add
#
# Example:
# addDictToDict({'a':1,'b':2}, {'b':1,'c':2}) = {'a':1,'b':4,'c':2}
#
#-----------------------------------------------------------------

def addDictToDict(dict1, dict2):
   for key in dict2:
      if key in dict1:
         dict1[key] += dict2[key]
      else:
         dict1[key] = dict2[key]

#-----------------------------------------------------------------
#
# subtractDictFromDict - subtract a dictionary with int value from another
dictionary
#
# Input: dict1 - operand1, the one that is modified to
#        dict2 - operand2, the one to substract
#
# Example:
# subtractDictFromDict({'a':1,'b':2}, {'b':1,'c':2}) = {'a':1,'b':1,'c':-2}
#
#-----------------------------------------------------------------

def subtractDictFromDict(dict1, dict2):
   for key in dict2:
      if key in dict1:
         dict1[key] -= dict2[key]
      else:
         dict1[key] = -dict2[key]

#=================================================================
#
```

```
# CRawSentenceReader - the raw sentence reader
#
# This reader is aimed for Chinese.
#
#=================================================================

class CRawSentenceReader(object):

   #-----------------------------------------------------------------
   #
   # __init__ - initialisation
   #
   # Inputs: sPath - the file for reading
   #
   #-----------------------------------------------------------------

   def __init__(self, sPath, sEncoding="utf-8"):
      self.m_sPath = sPath
      self.m_oFile = open(sPath)
      self.m_sEncoding = sEncoding

   #-----------------------------------------------------------------
   #
   # __del__ - destruction
   #
   #-----------------------------------------------------------------

   def __del__(self):
      self.m_oFile.close()

   #-----------------------------------------------------------------
   #
   # readNonEmptySentence - read the next sentence
   #
   # Returns: list of characters or None if the EOF symbol met.
   #
   #-----------------------------------------------------------------

   def readNonEmptySentence(self):
      # 1. read one line
      sLine = "\n"                              # use a pseudo \n to start
      while sLine:                              # while there is a line
         sLine = sLine.strip()                  # strip the line
         if sLine:                              # if the line isn't empty
            break                               # break
         sLine = self.m_oFile.readline()        # read next line
         if not sLine:                          # if eof symbol met
            return None                         # return
```

```
      # 2. analyse this line
      uLine = sLine.decode(self.m_sEncoding)    # find unicode
      lLine = [sCharacter.encode(self.m_sEncoding) for sCharacter in uLine]
      return lLine

   #-------------------------------------------------------------
   #
   # readSentence - read the next sentence
   #
   # Returns: list of characters or None if the EOF symbol met.
   #
   #-------------------------------------------------------------

   def readSentence(self):
      # 1. read one line
      sLine = self.m_oFile.readline()           # read next line
      if not sLine:                             # if eof symbol met
         return None                            # return
      # 2. analyse this line
      uLine = sLine.strip().decode(self.m_sEncoding)    # find unicode
      lLine = [sCharacter.encode(self.m_sEncoding) for sCharacter in uLine]
      return lLine

#=============================================================
#
# CPennTaggedSentenceReader - the tagged sentence reader
#
#=============================================================

class CPennTaggedSentenceReader(object):

   #-------------------------------------------------------------
   #
   # __init__ - initialisation
   #
   # Inputs: sPath - the file for reading
   #
   #-------------------------------------------------------------

   def __init__(self, sPath):
      self.m_sPath = sPath
      self.m_oFile = open(sPath)

   #-------------------------------------------------------------
   #
   # __del__ - destruction
   #
   #-------------------------------------------------------------
```

```
   def __del__(self):
      self.m_oFile.close()

   #--------------------------------------------------------------
   #
   # readNonEmptySentence - read the next sentence
   #
   # Input: bIgnoreNoneTag - ignore _-NONE- tagged word?
   #
   # Returns: list of word, tag pairs or None if the EOF symbol met.
   #
   #--------------------------------------------------------------

   def readNonEmptySentence(self, bIgnoreNoneTag):
      # 1. read one line
      sLine = "\n"                              # use a pseudo \n to start
      while sLine:                              # while there is a line
         sLine = sLine.strip()                  # strip the line
         if sLine:                              # if the line isn't empty
            break                               # break
         sLine = self.m_oFile.readline()        # read next line
         if not sLine:                          # if eof symbol met
            return None                         # return
      # 2. analyse this line
      lLine = sLine.strip().split(" ")
      lNewLine = []
      for nIndex in xrange(len(lLine)):
         tTagged = tuple(lLine[nIndex].split("/"))
         if len(tTagged) >= 3:
            tTagged = ('/'.join(tTagged[:-1]), tTagged[-1])
         assert(len(tTagged)<3)
         if len(tTagged)==1:
            tTagged = (tTagged[0], "-NONE-")
         if (bIgnoreNoneTag==False) or (tTagged[0]): # if we take -NONE- tag, or
if we find that the tag is not -NONE-
            lNewLine.append(tTagged)
      return lNewLine

   #--------------------------------------------------------------
   #
   # readNonEmptySentence - read the next sentence
   #
   # Input: bIgnoreNoneTag - ignore _-NONE- tagged word?
   #
   # Returns: list of word, tag pairs or None if the EOF symbol met.
   #
   #--------------------------------------------------------------
```

```
   def readSentence(self, bIgnoreNoneTag):
      # 1. read one line
      sLine = self.m_oFile.readline()           # read next line
      if not sLine:                             # if eof symbol met
         return None                            # return
      # 2. analyse this line
      lLine = sLine.strip().split(" ")
      lNewLine = []
      for nIndex in xrange(len(lLine)):
         tTagged = tuple(lLine[nIndex].split("/"))
         assert(len(tTagged)<3)
         if len(tTagged)==1:
            tTagged = (tTagged[0], "-NONE-")
         if (bIgnoreNoneTag==False) or (tTagged[0]): # if we take -NONE- tag, or
if we find that the tag is not -NONE-
            lNewLine.append(tTagged)
      return lNewLine

#================================================================
#
# Global.
#
#================================================================

g_sInformation = "\nThe evaluation program for Chinese Tagger. \n\n\
  Yue Zhang 2006\n\
  Computing laboratory, Oxford\n\n\
evaluate.py candidate_text reference_text\n\n\
The candidate and reference text need to be files with tagged sentences. Each
sentence takes one line, and each word is in the format of Word_Tag.\n\n\
"

#----------------------------------------------------------------
#
# evaluateSentence - evaluate one sentence
#
# Input: tCandidate - candidate sentence
#        tReference
#
# Return: int for correct words
#
#----------------------------------------------------------------

def evaluateSentence(lCandidate, lReference):
   nCorrectWords = 0
   nCorrectTags = 0
   nChar = 0
```

```
   indexCandidate = 0
   indexReference = 0
   while lCandidate and lReference:
      if lCandidate[0][0] == lReference[0][0]:  # words right
         nCorrectWords += 1
         if lCandidate[0][1] == lReference[0][1]: # tags
            nCorrectTags += 1
         indexCandidate += len(lCandidate[0][0]) # move
         indexReference += len(lReference[0][0])
         lCandidate.pop(0)
         lReference.pop(0)
      else:
         if indexCandidate == indexReference:
            indexCandidate += len(lCandidate[0][0]) # move
            indexReference += len(lReference[0][0])
            lCandidate.pop(0)
            lReference.pop(0)
         elif indexCandidate < indexReference:
            indexCandidate += len(lCandidate[0][0])
            lCandidate.pop(0)
         elif indexCandidate > indexReference:
            indexReference += len(lReference[0][0]) # move
            lReference.pop(0)
   return nCorrectWords, nCorrectTags

#================================================================
#
# Main.
#
#================================================================

if __name__ == '__main__':
   #
   # Parse command …
   #
   opts, args = getopt.getopt(sys.argv[1:], "")
   for opt in opts:
      print opt
   if len(args) != 2:
      print g_sInformation
      sys.exit(1)
   sCandidate = args[0]
   sReference = args[1]
   if not os.path.exists(sCandidate):
      print "Candidate file %s does not exist." % sCandidate
      sys.exit(1)
   if not os.path.exists(sCandidate):
      print "Reference file %s does not exist." % sReference
```

```
      sys.exit(1)
   #
   # Compare candidate and reference
   #
   nTotalCorrectWords = 0
   nTotalCorrectTags = 0
   nCandidateWords = 0
   nReferenceWords = 0
   fReference = CPennTaggedSentenceReader(sReference); fCandidate =
CPennTaggedSentenceReader(sCandidate)
   lReference = fReference.readNonEmptySentence(bIgnoreNoneTag=True); lCandidate
= fCandidate.readNonEmptySentence(bIgnoreNoneTag=True)
   while lReference and lCandidate:
      n=len(lCandidate)
      nCandidateWords += len(lCandidate)
      nReferenceWords += len(lReference)
      nCorrectWords, nCorrectTags = evaluateSentence(lCandidate, lReference)
      nTotalCorrectWords += nCorrectWords
      nTotalCorrectTags += nCorrectTags
      lReference = fReference.readNonEmptySentence(bIgnoreNoneTag=True);
lCandidate = fCandidate.readNonEmptySentence(bIgnoreNoneTag=True)

   if ( lReference and not lCandidate ) or ( lCandidate and not lReference ) :
      print "Warning: the reference and the candidate consists of different
number of lines!"

   word_precision = float(nTotalCorrectWords) / float(nCandidateWords)
   word_recall = float(nTotalCorrectWords) / float(nReferenceWords)
   tag_precision = float(nTotalCorrectTags) / float(nCandidateWords)
   tag_recall = float(nTotalCorrectTags) / float(nReferenceWords)
   word_fmeasure = (2*word_precision*word_recall)/(word_precision+word_recall)
   if tag_precision+tag_recall==0:
      tag_fmeasure = 0.0
   else:
      tag_fmeasure = (2*tag_precision*tag_recall)/(tag_precision+tag_recall)

   print "Tag precision:", tag_precision
```

[100]:
```
!python2.7 ./script/evaluate.py ./closed.hyp ./data/corpus.clean
```

```
Tag precision: 1.0
```

လက်တွေ့ open test နဲ့ decoding လုပ်တဲ့အခါမှာ unknown word တွေပါတဲ့ လိုင်းတွေကို ဆက်လက် word segmentation မလုပ်တာမို့လို့ reference နဲ့ hypothesis ဖိုင်နှစ်ဖိုင်ရဲ့ စာကြောင်းရေ အရေအတွက်မတူတဲ့အခါမျိုးလည်း ဖြစ်တတ်ပါတယ်။ အဲဒီလိုအခြေအနေမှာ တွက်ပေးနိုင်ဖို့အတွက် evaluate_segmentation.py code ကို ရေးခဲ့တယ်။

[101]:
```
!python ./script/evaluate_segmentation.py --help
```

```
usage: evaluate_segmentation.py [-h] --ref REF --hyp HYP [--top-k TOP_K]

Boundary-based evaluation for word segmentation (Precision, Recall, F1 +
Token-level Top-K Errors)

options:
  -h, --help     show this help message and exit
  --ref REF      Path to reference (gold) word segmented file
  --hyp HYP      Path to hypothesis file with line ID and segmented output
  --top-k TOP_K  Top-K most frequent token-level segmentation errors to report
                 (default: 10)
```

1.16 Let's Learn evaluate_segmentation.py Code

ဆရာကိုယ်တိုင်ရေးထားတဲ့ evaluate_segmentation.py code ကိုလည်း လေ့လာကြည့်ပါ။

[113]:
```
!cat ./script/evaluate_segmentation.py
```

```
"""
Written by Ye Kyaw Thu, LU Lab., Myanmar.
Last updated: 11 July 2025
Usage:
    python ./script/evaluate_segmentation.py --ref data/otest.nopipe.word --hyp
open_test.hyp --top-k 10 > open_tesst_score.txt
    python ./script/evaluate_segmentation.py --ref data/ctest10.nopipe.word
--hyp closed_test.hyp --top-k 10 > closed_test_score.txt
"""

import argparse
from collections import Counter
import sys

def parse_args():
    parser = argparse.ArgumentParser(
        description="Boundary-based evaluation for word segmentation (Precision,
Recall, F1 + Token-level Top-K Errors)"
    )
    parser.add_argument("--ref", type=str, required=True,
                        help="Path to reference (gold) word segmented file")
    parser.add_argument("--hyp", type=str, required=True,
                        help="Path to hypothesis file with line ID and segmented
output")
    parser.add_argument("--top-k", type=int, default=10,
                        help="Top-K most frequent token-level segmentation
errors to report (default: 10)")
    return parser.parse_args()

def load_reference(ref_path):
```

```
    ref_dict = {}
    with open(ref_path, 'r', encoding='utf-8') as f:
        for i, line in enumerate(f):
            tokens = line.strip().split()
            ref_dict[i] = tokens
    return ref_dict

def load_hypothesis(hyp_path):
    hyp_dict = {}
    with open(hyp_path, 'r', encoding='utf-8') as f:
        for line in f:
            if '|||' not in line:
                continue
            idx_str, hyp = line.strip().split('|||', maxsplit=1)
            try:
                idx = int(idx_str.strip())
                tokens = hyp.strip().split()
                hyp_dict[idx] = tokens
            except ValueError:
                continue
    return hyp_dict

def get_boundaries(tokens):
    """Return set of character offset boundaries for token list"""
    boundaries = set()
    offset = 0
    for token in tokens[:-1]:  # No boundary after last token
        offset += len(token)
        boundaries.add(offset)
    return boundaries

def extract_token_level_errors_by_spans(ref_tokens, hyp_tokens):
    errors = []

    # Reconstruct original sentence
    ref_text = "".join(ref_tokens)
    hyp_text = "".join(hyp_tokens)

    if ref_text != hyp_text:
        # Can't compare if character sequences mismatch (e.g., decoding issues)
        return errors

    def get_spans(tokens):
        spans = []
        offset = 0
        for token in tokens:
            start = offset
            end = offset + len(token)
```

```
            spans.append((start, end, token))
            offset = end
        return spans

    ref_spans = get_spans(ref_tokens)
    hyp_spans = get_spans(hyp_tokens)

    hyp_span_set = {(start, end) for (start, end, _) in hyp_spans}
    hyp_span_map = {(start, end): token for (start, end, token) in hyp_spans}

    for (start, end, gold_token) in ref_spans:
        if (start, end) not in hyp_span_set:
            # Find predicted tokens that overlap this gold span
            merged_pred = ""
            for (h_start, h_end, h_token) in hyp_spans:
                if h_start >= end:
                    break
                if h_end <= start:
                    continue
                merged_pred += h_token
            errors.append((gold_token, merged_pred))

    return errors

def evaluate(ref_dict, hyp_dict):
    TP, FP, FN = 0, 0, 0
    error_counter = Counter()

    for idx, ref_tokens in ref_dict.items():
        ref_bounds = get_boundaries(ref_tokens)
        hyp_tokens = hyp_dict.get(idx)

        if hyp_tokens is None:
            FN += len(ref_bounds)
            continue

        hyp_bounds = get_boundaries(hyp_tokens)

        TP += len(ref_bounds & hyp_bounds)
        FP += len(hyp_bounds - ref_bounds)
        FN += len(ref_bounds - hyp_bounds)

        # Track token-level errors
        for ref_tok, hyp_tok in extract_token_level_errors_by_spans(ref_tokens,
hyp_tokens):

            if ref_tok != hyp_tok:
                error_counter[(ref_tok, hyp_tok)] += 1
```

```
    precision = TP / (TP + FP) if (TP + FP) > 0 else 0
    recall    = TP / (TP + FN) if (TP + FN) > 0 else 0
    f1        = 2 * precision * recall / (precision + recall) if (precision +
recall) > 0 else 0

    return precision, recall, f1, error_counter

def main():
    args = parse_args()

    ref_dict = load_reference(args.ref)
    hyp_dict = load_hypothesis(args.hyp)

    precision, recall, f1, error_counter = evaluate(ref_dict, hyp_dict)

    print("\n=== Word Segmentation Evaluation ===")
    print(f"Total Ref lines: {len(ref_dict)}")
    print(f"Total Hyp lines: {len(hyp_dict)}\n")
    print(f"Precision: {precision:.4f}")
    print(f"Recall:    {recall:.4f}")
    print(f"F1 Score:  {f1:.4f}")

    print(f"\nTop-{args.top_k} Frequent Token-level Segmentation Errors:")
    for i, ((ref_tok, hyp_tok), count) in
enumerate(error_counter.most_common(args.top_k), 1):
        print(f"[{i}] Count: {count}")
        print(f"  REF token:     {ref_tok}")
        print(f"  Predicted as:  {hyp_tok}\n")

if __name__ == "__main__":
    main()
```

1.17 Prepare Open Test Data

ဒီ ဒီမိုအတွက် open test ဒေတာကိုတော့ graph ထုတ်ပြီး လေ့လာရ လွယ်ကူအောင် ဆောက်ထားတဲ့ “မမ ဝဝ” ကဗျာ corpus အသေးလေးပေါ်ကိုပဲ အခြေခံပြီး လက်နဲ့ပဲ ပြင်ဆင်ပါမယ်။

[114]:
```
!cat ./data/open_test.txt
```

မြမြ ဝဝ ထထ က
အက ပထမ
ကပါ ကပါ မြမြ ရာ
ညည လသာသာ
ညအခါ ငါ စာ မ ရ
မြမြ ဝဝ ထထ က

1.17.1 Editing Notes

အထက်မှာ မြင်ရတဲ့အတိုင်း “မမ” နေရာမှာ “မြမြ” နဲ့ အစားထိုးထားတယ်။
ပြီးတော့ “မြမြ ဝဝ” နဲ့ “ထထ က” ဆိုတဲ့ စာကြောင်းကို နှစ်ကြောင်းခွဲမထားပဲ ပေါင်းရေးပြီး စာကြောင်းတစ်ကြောင်း အဖြစ်ဖြစ် ထားထားတာမျိုးတွေလည်း ရှိတယ်။
“ငါ စာ မ ရ” ဆိုပြီးတော့ “စာရ” ဆိုတဲ့ စာလုံးရဲ့ အကြားမှာလည်း “မ” ထည့်တာမျိုး editing လုပ်ထားပါတယ်။
အဲဒါမှလည်း open test ရဲ့ nature အတိုင်း မော်ဒယ်ဆောက်တုန်းက မပါတဲ့ word တွေ ပါလာမှာမျိုးဖြစ်လို့ simulation လုပ်ယူထားတာပါ။

1.18 Character Segmentation

Character တစ်လုံးချင်းစီ ဖြစ်သွားပြီးမှ input လုပ်ပြီး စာလုံးဖြတ်ပေး မပေး (သို့) စာလုံးအဖြစ် တွဲပေး မပေး ကြည့်ချင်တာမို့ ...

[117]:
```
!cat ./data/open_test.txt | python ./script/char_break.py > ./data/open_test.
↪char
```

1.18.1 Check Character Segmented Output File

Character ဖြတ်ပြီးထွက်လာတဲ့ output ဖိုင်ကို ဝင်စစ်ဆေးကြည့်ရအောင်။

[118]:
```
!cat ./data/open_test.char
```

```
မ ြ မ ြ ဝ ဝ ထ ထ က
အ က ပ ထ မ
က ပ ါ က ပ ါ မ ြ မ ြ ရ ာ
ည ည လ သ ာ သ ာ
ည အ ခ ါ င ါ စ ာ မ ရ
မ ြ မ ြ ဝ ဝ ထ ထ က
```

1.19 Decoding with Open Test Data

တနည်းအားဖြင့် open test data ကိုသုံးပြီး word segmentation လုပ်ခိုင်းတာပါပဲ။
ဆောက်ထားတာက word segmentation FST မို့လို့ ပါ။

ဒီတခါတော့ မော်ဒယ်က ထွက်လာတဲ့ စာကြောင်းတွေက input file ရဲ့ စာကြောင်းအရေအတွက်အတိုင်း ဟုတ်ချင်မှ ဟုတ်မှာမို့ –show-id ဆိုတဲ့ option ကိုသုံးပြီးတော့ decode လုပ်ပါမယ်။

[123]:
```
!time python ./fst_decoder.py ./config.yaml  --show-id < ./data/open_test.char␣
↪> ./open.hyp
```

```
Initializing model with config: {'type': 'plain', 'file': './lexicon_final.fst',
'input_symbols': './data/char.sym', 'output_symbols': './data/word.sym',
'weights': [1.0]}
Model object created, attempting to load…
Attempting to load FST from: ./lexicon_final.fst
Successfully loaded FST with 10 states
```

```
Input symbols: yes
Output symbols: yes
Model loaded successfully. Start state: 0
Num states: 10
Input symbols: <const Fst SymbolTableView './data/char.sym' at 0x7dc5c438b7b0>
Output symbols: <const Fst SymbolTableView './data/word.sym' at 0x7dc5c438b7b0>
-- Starting FST Decoder --
Common symbols: 19
Composing with model: ./lexicon_final.fst
Composition failed - possible symbol mismatch
Input FST symbols: <const Fst SymbolTableView './data/char.sym' at
0x7dc5c438bc00>
Model FST symbols: <const Fst SymbolTableView './data/char.sym' at
0x7dc5c438bc00>
WARNING: no path found
Common symbols: 19
Composing with model: ./lexicon_final.fst
Common symbols: 19
Composing with model: ./lexicon_final.fst
Composition failed - possible symbol mismatch
Input FST symbols: <const Fst SymbolTableView './data/char.sym' at
0x7dc5c438b900>
Model FST symbols: <const Fst SymbolTableView './data/char.sym' at
0x7dc5c438b900>
WARNING: no path found
Common symbols: 19
Composing with model: ./lexicon_final.fst
Common symbols: 19
Composing with model: ./lexicon_final.fst
Composition failed - possible symbol mismatch
Input FST symbols: <const Fst SymbolTableView './data/char.sym' at
0x7dc5c438b900>
Model FST symbols: <const Fst SymbolTableView './data/char.sym' at
0x7dc5c438b900>
WARNING: no path found
Common symbols: 19
Composing with model: ./lexicon_final.fst
Composition failed - possible symbol mismatch
Input FST symbols: <const Fst SymbolTableView './data/char.sym' at
0x7dc5c438b900>
Model FST symbols: <const Fst SymbolTableView './data/char.sym' at
0x7dc5c438b900>
WARNING: no path found

real    0m0.057s
user    0m0.048s
sys     0m0.009s
```

1.20 Let's Check Segmented Open Data

FST model ကနေ ဖြတ်ပြီးထွက်လာတဲ့ output ဖိုင်ကို လေ့လာကြည့်ရင် အောက်ပါအတိုင်း တွေ့ရပါလိမ့်မယ်။

[124]:
```
!cat ./open.hyp
```

```
1|||အက ပထမ
3|||ည ည လ သာ သာ
```

Input ဖိုင် (character segmentation မဖြတ်ရသေးခင်က) နဲ့ ပြန်နှိုင်းယှဉ်ကြည့် ရအောင်...

[125]:
```
!cat ./data/open_test.txt
```

```
မြမြ ဝဝ ထထ က
အက ပထမ
ကပါ ကပါ မြမြ ရာ
ညည လသာသာ
ညအခါ ငါ စာ မ ရ
မြမြ ဝဝ ထထ က
```

1.21 Evaluation

ဒီတခါတော့ reference ဖိုင်နဲ့ hypothesis က လိုင်းအရေအတွက် မတူညီတဲ့ကြားထဲကနေပဲ evaluation အကြမ်းလုပ်ကြည့်ချင်တာမို့လို့ evaluate_segmentation.py code ကို သုံးပါမယ်။

[126]:
```
!python ./script/evaluate_segmentation.py --ref ./data/open_test.txt --hyp ./
↪open.hyp
```

```
=== Word Segmentation Evaluation ===
Total Ref lines: 6
Total Hyp lines: 2

Precision: 0.4000
Recall:    0.1333
F1 Score:  0.2000

Top-10 Frequent Token-level Segmentation Errors:
```

1.22 Summary

ဒီ တခေါက်က FST ရဲ့ အလုပ်လုပ်ပုံကို မြင်စေချင်လို့ ဆရာဦးတင်မိုးရဲ့ "မမ ဝဝ" ကလေးကဗျာလေးကိုပဲ corpus အသေးလေးဆောက်ခဲ့ပြီး Lexicon FST ကို visualization လုပ်ပြခဲ့တာဖြစ်ပါတယ်။ Corpus size က တအားသေးတော့ optimization မလုပ်ခင်နဲ့ လုပ်ပြီးသားမှာ filesize အနေနဲ့ အရမ်းကွာခြားသွားတာမျိုး မဟုတ်ပေမဲ့ graph အပြောင်းအလဲတွေကိုတော့ အတိုင်းအတာတခုထိ မြင်နိုင်တာမို့ အကျိုးရှိပါလိမ့်မယ်။

[ ]:
```

```