Natural Language Processing Homework 6 README

Zixuan Wang, Mou Zhang

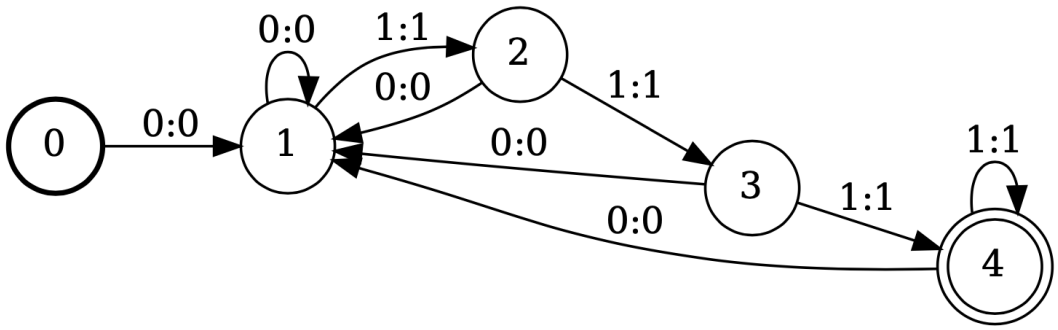
November 25, 2019

1 Question 1

(a) i If the typed string matches the FST, it will print itself. If not, it will print "Rewrite failed". .

ii First will accept any binary string that starts with 1 zero and ends with 3 ones. You can put any numbers of zeros or ones between the 1 zero and 3 ones.
Basic string FSTs are defined by text enclosed by double quotes. 0s and 1s should be enclosed by double quote because they are input strings instead of numerical values.

iii There are 5 states and 9 arcs.



(b) i In this step, we added Second and Disagreements to the binary.grm.

ii Disagreements should not accept any string and any input string will prompt "Rewrite failed". We can conclude this by showing that the state and arc of Disagreements are both 0, which indicates that it doesn't accept any input string.

```
fst type                                vector
arc type                                standard
input symbol table                      **Byte symbols
output symbol table                     **Byte symbols
# of states                             0
# of arcs                              0
initial state                           -1
# of final states                      0
# of input/output epsilons             0
# of input epsilons                    0
# of output epsilons                   0
input label multiplicity                0
output label multiplicity               0
# of accessible states                  0
# of coaccessible states                0
# of connected states                   0
# of connected components               0
# of strongly conn components           0
input matcher                           y
output matcher                          y
input lookahead                         n
output lookahead                        n
expanded                                y
mutable                                 y
error                                   n
acceptor                                y
input deterministic                     y
output deterministic                    y
input/output epsilons                   n
input epsilons                          n
output epsilons                         n
input label sorted                      y
output label sorted                     y
weighted                                n
cyclic                                  n
cyclic at initial state                 n
top sorted                              y
accessible                              y
coaccessible                            y
string                                  y
weighted cycles                         n
```

(c) i Fist will have 20 states and 25 arcs. Second will have 13 states and 16 arcs. Disagreement will have 88 states and 113 arcs.

ii Because in the expression, it has (First - Second) and (Second - First), there should be 2 branches corresponding to those 2 expressions in order to calculate separately. In the end, the expression Union those two branches together, which is why they merged in the end.

iii The result should be the same with the optimized version. Because the only difference between optimized and unoptimized is the ϵ transaction. This transaction will not make any change to the result of the output because people cannot type ϵ in the query.

(d) We get 0 state and 0 arc for optimizing Disagreements which is the same as doing optimize after optimize First and Second. The reason why we get this because optimize works really well in reducing all the states. When no possible final state exists, it will generate 0 state and 0 arc in the end.

2 Question 2

(a) export Triplets = Optimize [(Zero* One One One One* Zero*)];

```
Input string: 111111
Output string: 111111
Input string: 01010111
Rewrite failed.
Input string: 011110010011100111
Rewrite failed.
Input string: 01111011110111011
Rewrite failed.
Input string: 111110111101111011111011110
Output string: 111110111101111011111011110
Input string:
```

(b) export NotPillars = Optimize [Bit* - (One One)*];

```
Input string: 11
Rewrite failed.
Input string: 1111
Rewrite failed.
Input string: 100101
Output string: 100101
Input string: 0101
Output string: 0101
Input string: 0101011
Output string: 0101011
Input string: 010101
Output string: 010101
Input string: 11
Rewrite failed.
Input string: 1111
Rewrite failed.
Input string: 1101011
Output string: 1101011
Input string: 111111
Rewrite failed.
Input string:
```

(c) export Oddlets = Optimize[(Zero*) — Zero* One (One One)* (Zero+ One (One One)*)* Zero*];

```
[mzhan106@ugrad12 assignment6]$ grmtest binary.grm Oddlets
make: 'binary.far' is up to date.
Input string: 10101001
Output string: 10101001
Input string: 010110101011
Rewrite failed.
Input string: 1111111
Output string: 1111111
Input string: 01010101
Output string: 01010101
Input string: 101010101110001111100
Output string: 101010101110001111100
Input string: 101010101
Output string: 101010101
Input string: 01010110
Rewrite failed.
Input string: 0000
Output string: 0000
Input string: 11111
Output string: 11111
Input string:
```

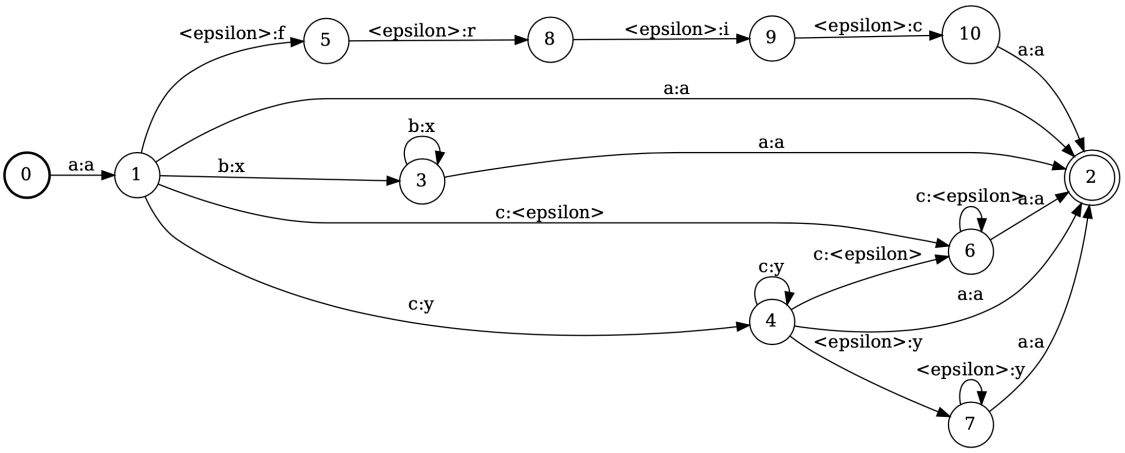
3 Question 3

(a) The input language should be a(b*|c+| ϵ)a

(b) 0 output: zzzzz
1 output: aba
2 output : aa
zz more than 2 outputs: aca

(c) We don't need to answer this question.

(d) Yes, it consists with our answer above. There are 11 states and 20 arcs in the optimized version of Cross.



4 Question 4

(a) Please see BitFlip1 in rewrite.grm

(b) Please see BitFlip2 in rewrite.grm

(c) We don't think ϵ is a valid binary number. We treat ϵ as neither an even number nor an odd number because in real life it is neither of them.

(d) Please see Parity2 in rewrite.grm

(e) If we use CDRewrite, it will treat ϵ as an empty string because Bit* would contain empty string and it is between [BOS] and [EOS]. It is a valid input and produce an empty string as output.

(f) This FST would only accept 2 inputs: "0" and "1". If you type "0", it will generate many binary even numbers and if you type "1", it will generate many binary odd numbers. If you type ϵ, it will treat it as an empty string and produce an output as empty string.

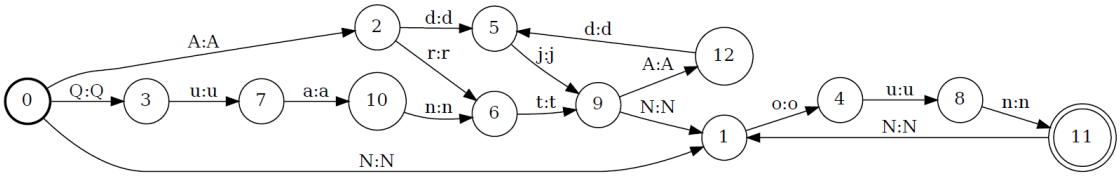
(g) Please see Split in rewrite.grm

(h) Please see SplitThree in rewrite.grm

5 Question 5

(a) i You should put double quotation marks on those tags. The fixed version should be the following one: export NP = Optimize [("Art"|"Q5uant")? ("Adj")* ("Noun")+];
It will accept strings like "ArtAdjNoun", "QuantAdjNoun" and "AdjNounNoun" etc.

ii There are 13 states and 17 arcs for NP.fst. The machine use every single character as the input between the states and reuse the same character for different word. For exmaple, "t" is used for both "Art" and "Quant".



(b)

(c) i This composition is taking everything that satisfies NP and use it to do MakeNmod to get the result. If the initial input is not NP, it will fail the transducer.

ii The result of ArtAdjNounNounNoun is ArtAdjNmodNmodNoun
The input AdjNounNounNounVerb will fail the transducer.

iii There are 16 states and 20 arcs in TransformNP. There are 25 states and 40 arcs in MakeNmod.

iv The topology of TransformNP is pretty similar to the topology of NP. The only difference is that in TransformNP, there is one more step to change every non-last Noun to Nmod. It is like a loop before reaching the final Noun.

(d) Brackets1 will show results which allow to leave some replaceable substrings unreplaced, which would usually have more results in the end.
Brackets2 has a obligatory ('obl') mode which means unreplaced regions may not contain any more replaceable substrings, which would usually have less results in the end.

(e) Please see BracketTransform in chunker.grm

(f) Please see BracketResults in chunker.grm

(g) Please see TranformNP2 in chunker.grm

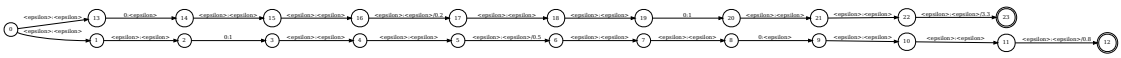
6 Question 6

(a) i The minimum-weight string accepted by this FSA is ZeroBitOne and its weight is 1 + 0.2 + 0.5 = 1.7.

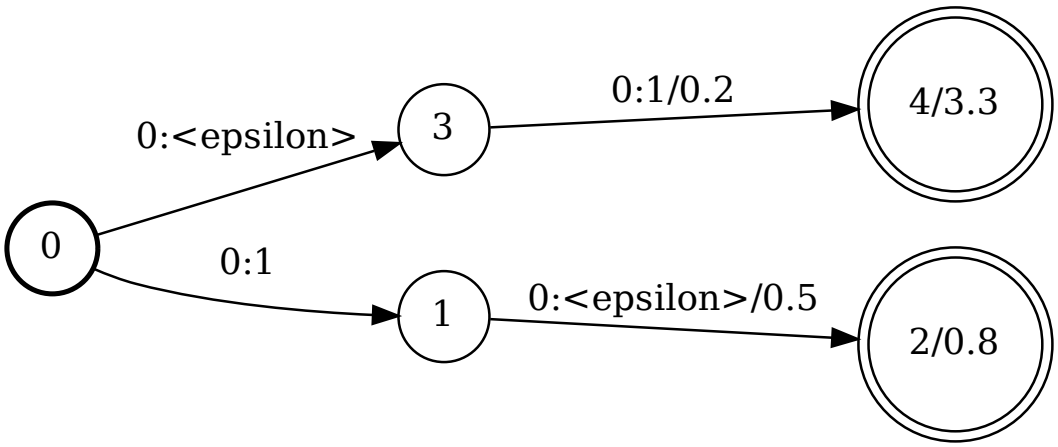
ii The minimum-weight pair of strings should also be ZeroBitOne and its weight should be 1 + 0.2 + 0.75 = 1.95.

(b) Please see WFlip in Binary.grm

(c) i We used (x,y) = (00, 1) as an exmaple. Out FST only takes 00 as input and 1 as output. We randomly assign different weights to those two different paths and finally get the minimized result. The weight of the first paths is 3.5 and the weight of the second path 1.3. Please see the diagram for more information.



ii There are two paths accept (x,y) and their weights are the same as before. The first one is 3.5 and the second one is 1.3. Optimize only helps us reduce those ϵ between states, which should return the same result as before. Please see the following diagrams for more information.



iii If you do (””:T_out), it will give you any possible outputs of T, no matter what input it is. If you do (””:xT_out;) will give you any possible output from T where input is x. If you do (””:Ty_in;) it will give you any possibile input of T where output is y. If you want to know the mimimal cost of the path, you can use export the exTye_opt and give it ϵ by typing Enter.

The last FSM will have 1 state and 0 arc.

xT_out_opt is important because it will keep all the possible outputs of the corresponding path with the minimal weight.

yT_in_opt is important because it will keep all possible inputs of the path with minimum weight.

exTye_opt is important because it will give the minimum weight of T.

(d) Please see NoDet in binary.grm

7 Question 7

(a) wc -w entrain.txt will show 95936 entrain.txt
wc -l entrain.alpha will show 12410 entrain.alpha
fstinfo entrain.fst will show the following

fst type vector
arc type standard
input symbol table entrain.sym
output symbol table entrain.sym
# of states 68479
# of arcs 217386
initial state 1
# of final states 2016
# of input/output epsilons 68478
# of input epsilons 68478
# of output epsilons 68478
input label multiplicity 1
output label multiplicity 1
# of accessible states 68479
# of coaccessible states 68479
# of connected states 68479
# of connected components 1
# of strongly conn components 915

input matcher y
output matcher y
input lookahead n
output lookahead n
expanded y
mutable y
error n
acceptor y
input deterministic y
output deterministic y
input/output epsilons y
input epsilons y
output epsilons y
input label sorted y
output label sorted y
weighted y
cyclic y
cyclic at initial state n
top sorted n
accessible y
coaccessible y
string n
weighted cycles y

(b) ngramrandgen –max_sents=1 –remove_epsilon entrain.fst |farprintstrings will randomly generate some sentences like "Mr. the copper cornfield that the can Federal lawyers and rush to 21 contributed is attributed the decline acts will be able to junk . rose at in 1988 respected appointment 457.52 especially disaster loans . " which would usually have around 30 words in the sentence. The length is quite longer.

fstprintstring entrain.fst will generate relatively shorter sentences compared with previous one, usually around 5 words.
However, fstshortestpath entrain.fst | fstprintstring will always generate the shortest string which is ".". It only has a period in the end.

The main reason behind this is that in ngramrandgen uses ngram model with backoff method. It would be possible to make a word that is not in the training data appear and therefore it is much longer.

For fstprintstring, it is more likely to pick some states in the FST and do some transitions to reach the final state. There is no backoff method in this case, so it is shorter than the previous one.

(c) Nothing to hand in.

(d) The output of "Andy cherished the barrels each house made ." is
Andy cherished the barrels each house made . $< cost = 67.428474 >$

The output of "If only the reporters had been nice ." is
If only the reporters had been nice . $< cost = 49.260509 >$

There is no output for "Thank you".

We only have one result for each sentence which we assume it has the minimum cost of that sentence. The reason why we don't have any result for "Thank you" is that "Thank" is not in the domain of the corpus.

The domain and range of relation LM is entrain.sym because the transducer will not accept anything that is not in entrain.sym.

(e) The word cost from CompleteWord should be the same as the unigram probability of the word.

8 Question 8

(a) Please see noisy.grm

(b) Please see noisy.grm

(c) Please see noisy.grm

(d) The results are shown as following:

(1). Input string: Ifonlythereporterhadbeennice.
Output string: If only the reporter had been nice . $< cost = 50.265888 >$

(2). Input string: If only..

Rewrite failed.

(3). Input string: ThereportersaidtothecitythatEveryoneIskilled.

Output string: The reporters aid to the city that Everyone Is killed . $< cost = 70.201134 >$
Output string: The reporter said to the city that Everyone Is killed . $< cost = 70.476219 >$
Output string: The reporters aid to the city that Everyone I skilled . $< cost = 73.106644 >$
Output string: The reporter said to the city that Everyone I skilled . $< cost = 73.381729 >$
Output string: The reporters aid to the city that Every one Is killed . $< cost = 77.453484 >$
Output string: The reporter said to the city that Every one Is killed . $< cost = 77.728569 >$
Output string: The reporters aid to the city that Every one I skilled . $< cost = 80.358986 >$
Output string: The reporter said to the city that Every one I skilled . $< cost = 80.634071 >$

(4). Input string: Thankyou.

Rewrite failed.

(e)
i w_1 is the cost of adding another character to existing characters, which is 0.1 (the negative log of 0.9) and w_2 is the cost of generating a word whose length is at least 1, which is 2.3 (the negative log of 0.1).

ii $p_n = 0.1 * (0.9^{n-1})$ when n > 0 and $p_n = 0$ when n = 0.

iii $\sum_{n=0}^{\infty} p_n = \sum_{n=1}^{\infty} 0.1 \times 0.9^{n-1} = 1.0$

iv Decreasing w1 and increasing w2.

v The probability to get more words increases. This will increase the length of words, which makes the decoding process slower than before. There are more words need to be decoded.

vi According to our research, the average length of English words is 4.7. We can use this number to update w_1 and w_2. Or we can use bigram or trigram model on characters to improve those probability models.

(f) We run the code with the following command:

grmtest-with-symbols noisy.grm Decode2 byte entrain.sym 5

In this case, we limit the result to be 5 outputs.
The result on those 4 sentences would be following:

(1). Input string: Ifonlythereporterhadbeennice.

Output string: If only the reporter had been nice . $< cost = 50.265888 >$
Output string: $< unk >$ only the reporter had been nice . $< cost = 64.236694 >$
Output string: I $< unk >$ only the reporter had been nice . $< cost = 65.134201 >$
Output string: If only the report $< unk >$ had been nice . $< cost = 65.182625 >$
Output string: If on $< unk >$ the reporter had been nice . $< cost = 67.337349 >$

(2). Input string: If only.
Rewrite failed.

(3). Input string: ThereportersaidtothecitythatEveryoneIskilled.
Output string: The reporters aid to the city that Everyone Is killed . $< cost = 70.201134 >$
Output string: The reporter said to the city that Everyone Is killed . $< cost = 70.476219 >$
Output string: The reporters aid to the city that Everyone I skilled . $< cost = 73.106636 >$
Output string: The reporter said to the city that Everyone I skilled . $< cost = 73.381721 >$
Output string: The reporters aid to the city that Every one Is killed . $< cost = 77.453476 >$

(4).Input string: Thankyou.
Output string: $< unk >$ you . $< cost = 43.265034 >$
Output string: $< unk >$ an $< unk >$ you . $< cost = 48.048958 >$
Output string: $< unk >$. $< cost = 49.837250 >$
Output string: $< unk >$ a $< unk >$ you . $< cost = 52.368877 >$
Output string: $< unk >$ $< unk >$ you . $< cost = 53.026382 >$

(g) One obvious error in the recovered file is that it tends to split the long number into several numbers with smaller digits. For example, it tends to split 2.9428 to 2.9 4 28. In general, number with smaller digits will be more popular in the model. One way to deal with this problem is to assign proper weight to some long numbers.

Another error is that the recovered file cannot recognize some ProperNoun like Anton, it will split it into An ton. To address this problem, we should find a better way to smooth the words with lower probability.

(h) Total edit distance 713 over 50 lines (about 14 per line). The largest edit distance is 66 and the smallest edit distance is 0.

9 Question 9

i We design the following FST to transduce lowercase letter to digit on telephone keyboard.
$LowerToDigits = Optimize[(("a"|"b"|"c"|"2") : "2") < 1 > |(("d"|"e"|"f"|"3") : "3") < 1 > |(("g"|"h"|"i"|"4") : "4") < 1 > |(("j"|"k"|"l"|"5") : "5") < 1 > |(("m"|"n"|"o"|"6") : "6") < 1 > |(("p"|"q"|"r"|"s"|"7") : "7") < 1 > |(("t"|"u"|"v"|"8") : "8") < 1 > |(("w"|"x"|"y"|"z"|"9") : "9") < 1 >];$
Please see the LowerToDigits in noisy.grm for more readable version.

ii We implemented the FST based on the actual keyboard setup of the mobile phone. The reason why we design this way because that is the way cellphone is designed. We assign each weight to be 1 because we assume that those buttons would have the same probability to be pressed.

iii We use the following method to test the result:

entest.txt would the following
despite the increase , the british currency slid below a perceived threshold of three marks early last week .

entest-noisy.txt would be the following
3377483 843 46273273 843 2748474 28773629 7543 23569 2 737234833 847374653 63 84733 62757 32759 5278 9335

iv grmfilter -r noisy.grm Convert ¡ entest.txt ¿ mytest-noisy.txt

v It is almost impossible to to translate the version back to the previous one because we don't actually have the correct weights for all the input string. We do try our best to change the character to the actual number on the key board. But when we work backwards, things get a little bit complicated.

Use the following command to generate the recovered txt file:
grmfilter noisy.grm DigitsToLower ¡ mytest-noisy.txt ¿ mytest-recovered.txt

vi editdist entest.txt mytest-recovered.txt

Total edit distance 5647 over 50 lines (about 112 per line)