Natural Language Processing Homework 7

Katie Chang + Willis Wang

April 29, 2016

README

1 Q2

1.1 first

First accepts a language that starts with at least one 0 and ends with at least three 1s. There can be any combination of 0s and 1s in between the leading 0 and the trailing 111. Since the ? denotes optionality, the fourth 1 is not necessary for the string to be accepted in this language.

ii. This language is a subset of the language Bit*.

The 0 and 1 are quoted in the .grm file because these two FSMs are defining specific strings that this language will use. Essentially, the quotes indicate the initializing of the letters of the alphabet that this language will use.

iii. There are 5 states and 9 arcs in this FST.

1.2 b. second

i. export Second = Optimize[Zero Bit* One One One];

ii. If First and Second are equivalent, then Disagreements should accept nothing.

Running fstinfo on Disagreements.fst, we can see that this FST has 0 states and 0 arcs, and in fact has a 0 count of everything. The initial state value is -1. Therefore, it should be unable to accept any sort of string.

1.3 c

i. First.fst now has 20 states and 25 arcs. Second.fst now has 13 states and 17 arcs. Disagreements.fst has 68 states and 88 arcs (!!!).

ii. The two separate sub-FSAs is because the Disagreement FST is taking the union of two FSTs, First - Second and Second - First.

iii. The results are the same. The only difference there is is the size of the FST and topology.

1.4 d

After creating Disagreements.fst, the number of states and arcs is the same as before. This shows that it doesn't matter whether or not we optimize the sub-FSTs, but there would be a lot of redundant work done. We should optimize if we want to avoid excessive overhead costs.

2 Q3

See binary.grm.

3 Q4

3.1 a

Input language: "a"("b"* — "c"+ — "")"a"

3.2 b

"abca" has no outputs.

"aba" has 1 output = "axa"

"aa" has 2 outputs = "aa" and "africa"

"aca" has more than 2 outputs.

3.3 c

Takes all strings that start and end with one a, as well as one of the following criteria:

1. replaces all "b"s between the two "a"s with "x"s
2. replaces all "c"s between the "a"s with any number of "y"s
3. replaces any "" between the "a"s with either the empty string or "fric".

3.4 d

Consistent. 10 states and 16 arcs.

4 5

4.1 c. Parity1

This particular transducer does not take into account the empty string.

4.2 f. UnParity

UnParity inverts the transduction. The only acceptable strings are "0" and "1" / any number of 0s before one "1". The former refers to even binary numbers, and the output lists a bunch of random ones. The latter refers to odd binary numbers, and will list a bunch of them.

5 6

5.1 a

i. With the given NP definition, the issue is that the strings are not in quotes. The program throws an error "Undefined symbol: Art". What is accepted: The list of strings that optionally start with "Art" or "Quant", has 0 or more instances of "Adj", and 1 or more instance of "Noun".

ii. There are 13 states and 17 arcs. "Adj" is represented as three separate letters in the FST, and not as a unit.

5.2 b

export MakeNmod = CDRewrite["Noun":"Nmod", "", "Noun", SigmaStar, 'sim', 'obl'];

5.3 c

i. This composition accepts only NP strings, and of those input strings, converts them with the MakeNmod rewrite FST.

ii. ArtAdjNounNounNoun -¿ ArtAdjNmodNmodNoun

AdjNounNounNounVerb -¿ Rewrite failed.

The second input failed because the input string is not in the NP form of (Adj*Noun+).

iii. TransformNP has 16 states and 20 arcs. MakeNmod has 36 states and 2331 arcs. This makes sense, since TransformNP accepts a smaller set of strings.

iv. The two FSTs are similar. At the end of these FSTs, NP accepts at least one ”Noun”. TransformNP instead takes these ”Noun”s and turns them into ”Nmod”, except for the last one.

5.4 d

Bracket1 divides NP chunks in more ways than Bracket2 does. It will output all possible NP chunks. Bracket2 will simply go as far as possible to find the longest NP chunk.

5.5 e

export BracketTransform = CDRewrite[BracketNP @ MakeNmod, ””, ””, SigmaStar, ’sim’, ’obl’];

5.6 f

See chunker.grm.

6 7

6.1 a

Check out stress.grm!

6.2 b

Input string: ev'apor'ating Output string: ev'aporating Output string: ev'apor'ating

Input string: 'incomm'unic'ado Output string: 'incommunic'ado Output string: 'incomm'unic'ado Output string: incomm'unicado Output string: 'incommunicado Output string: 'incomm'unicado Output string: incommunicado Output string: incommunic'ado Output string: incomm'unic'ado

6.3 c

see stress.grm.

6.4 d

Making some assumptions, we assume that if "y" is followed by a consonant, then it is probably a vowel. If it is followed by a vowel, then it is probably a consonant.

7 8

7.1 a

Lead as a noun and read as a verb. They both end with the same ending, but don't sound the same.

7.2 b

For "academic", the mouth opens more and in a more wide sense. The tongue seems to remain low in the mouth for this "a" sound.

For "academy", the mouth opens less. The sound made is more like an "uh" sound, which means the tongue doesn't really move until the "d" sound.

7.3 c

Results language: the words printed are words that start with a strong sound and go on to end with a weaker sound. A Dacytl is when a long syllable is followed by two short syllables.

7.4 d

See dactyls.grm.

7.5 e

Used overleaf.com to generate LaTeX document.