Natural Language Processing HW 1

Prof. Liang Huang

Due on Canvas Monday April 17 at 11:59pm.
Each group (up to 3 members) only needs to submit one copy.

CN Y RD NGLSH WTHT VWLS? In this assignment, you will build a finite-state machine to automatically restore vowels to vowelless text. (In English, this may not be a very useful thing to do, but in languages like Arabic and Hebrew where vowels are regularly omitted (cf. **abjad**[1]), it is extremely useful.)

Before you begin

- Finish EX1 individually.
- (Optional) Read Sections 1 and 2 of "A Primer on Finite-State Software for Natural Language Processing" (`http://www.isi.edu/licensed-sw/carmel/carmel-tutorial2.pdf`). A brief overview of some of Carmel's command-line options (run `carmel` to see complete list):

`-si`	expect a string on stdin
`-l`	compose stdin onto the left of the named FSA/FST(s)
`-r`	compose stdin onto the right of the named FSA/FST(s)
`-O`	print only output labels, suppress input labels
`-I`	print only input labels, suppress output labels
`-k` *n*	list *k* sequences rather than the whole FSA/FST
`-b`	batch mode: process multiple input lines
`-WE`	suppress weights and empty labels in *k*-best lists

- Download `http://classes.engr.oregonstate.edu/eecs/spring2017/cs519-001/hw1/hw1-data.tgz`:

`vocab`	list of English words
`vocab.small`	shorter list (for testing purposes only)
`strings`	English sentences
`strings.bad`	English sentences with bad spelling
`eval.py`	script for measuring accuracy

1 Finite-state acceptors

1. I've created an example FSA `try.fsa` for sequences consisting of the three English words: `an`, `at`, and `age`, separated by an underscore between consecutive words, and nothing else. It was done using a simple prefix tree idea. You can test it like this:

   ```
   echo "A T _ A N _ A G E" | carmel -slibOWE try.fsa
   ```

   Based on this example, try to create (by hand) the FSA `small.fsa` for words in `vocab.small`. Include a drawing/illustration of this FSA in your PDF report.

2. Write a python program `make.py` to create an FSA `english.fsa` that accepts all strings consisting of English words (as defined by `vocab`) separated by one underscore between consecutive words, and nothing else. The command-line of your python program should be like this:

   ```
   cat vocab | python make.py > english.fsa
   ```

   (FYI my `make.py` has only 25 lines.) The FSA should be letter-based, not word-based: that is, transitions should be labeled with letters, not whole words. For example, it should accept:

[1] See `https://en.wikipedia.org/wiki/Abjad`.

```
T H I S _ I S _ A _ S T R I N G
```

(a) How many states and how many transitions does your FSA have? (Use `carmel -c english.fsa`.)

(b) Verify that your FSA accepts every line in `strings` and no lines in `strings.bad`. Show the output of Carmel on the first five lines of each file. You can use commands like:

```
cat strings | carmel -slibOWE english.fsa
```

Hint: you should have either ∼250k or ∼180k states (and ∼360k transitions or less). No need to minimize it.

2 Finite-state transducers

3. Create a FST in Carmel format called `remove-vowels.fst` that deletes all English vowels, preserving word boundary information. For the purposes of this assignment, vowels are defined to be members of {A, E, I, O, U}. For example, it should perform the following mappings:

```
Y O U _ A R E _ H E R E ↔ Y _ R _ H R
  R E A D _ A _ B O O K ↔ R D _ _ B K
```

(a) Draw your FST in your PDF report in enough detail for someone else to replicate it.

(b) Test your FST in the forward direction on `strings` with the following command:

```
cat strings | carmel -slibOEWk 1 remove-vowels.fst
```

Show the output on the first five lines, and save the whole output to a file `strings.novowels`.

(c) Test your FST in the backward direction with the following command:

```
echo "B L D N G" | carmel -sriIEWk 10 remove-vowels.fst
```

The `-k 10` option asks for up to 10 output strings. **Just list them.**

4. Now you can use backwards application of your FST to do vowel restoration.

(a) Run your vowel restorer on `strings.novowels`, using the following command:

```
cat strings.novowels | carmel -sribIEWk 1 remove-vowels.fst
```

Show the output on the first five lines, and save the whole output to a file `strings.restored`.

(b) Compute your vowel-restoration accuracy using the command:

```
python eval.py strings strings.restored
```

What was your accuracy?

(c) Why is the score so low?

Hint: should be around 1.3%.

3 Combining FSAs and FSTs

5. The vowel restorer you built in the previous part had a problem. Can you fix it by combining your FSA and FST?

(a) Describe how to combine your FSA and FST to improve vowel restoration.

(b) Implement your idea and test it on `strings.novowels`. What is your new accuracy?

(c) Are the results satisfactory? Why doesn't the machine do what a human would do?

Hint: should be around 30%.

6. How might your vowel restorer be further improved? Come up with at least one idea, and for each idea:

(a) Describe it in enough detail so that someone else could replicate it.

(b) Implement it and try it on `strings.novowels`.

(c) Report your new accuracy on `strings.novowels`. *You will be graded on your final accuracy.*

Hint: very easy to get above 90%.

Submit your code (`*.py`), your FSAs/FSTs (`*.fsa, *.fst`), your outputs (`strings.*`), and `report.pdf`.