CSCI 3907/6907: Introduction to Statistical NLP

Assignment #1

Instructions:

- ✓ This assignment is due on **Friday, September 20, 2019 by 11:59 pm**. Refer to course information about late submission policy. Late days are counted from 00:00 am onwards.
- ✓ Submit your solution in a zipped folder through blackboard.
- ✓ All code must compile and run to receive credit for coding parts. We won't do any debugging. You can code in Perl, Python, Java, or C++. **Perl/Python** are recommended.
- ✓ Include brief instructions on how to run your scripts.
- ✓ Include citations for any online resources used or group discussions.

Part I: Regex and Finite State Machines

Q1.[10 points] Write regular expressions for the following languages:

a) The set of all strings with two consecutive repeated words (e.g. "`Humbert Humbert`" and "`the the`")

b) The set of all strings from the alphabet `a,b` such that each `a` is immediately preceded **or** immediately followed by a `b`

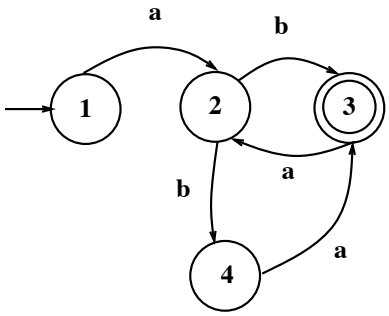
c) Dates in the form `mm/dd/yy`. Make sure months and days are in the valid range (1-12 for months, 1-31 for days), but you don't need to specify agreement (for example, `02/31/98` is still fine).

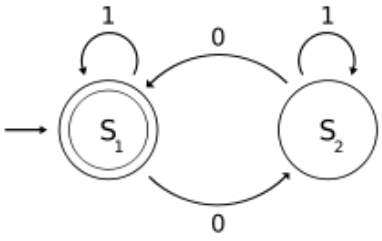
d) All strings that start at the beginning of a line with an integer, and end at the end of the line with a word.

e) All inflections of the verbs "`walk`", "`talk`", and "`break`".
"Note: the inflectional forms for break --> break breaks broke broken breaking"

Q2. [5 points] Draw a Finite State Automata (FSA) that corresponds to the languages in parts (b) and (c) above.

Q3. [5 points]Write the regular expressions that correspond to the following FSAs





Q4. [Required for graduate students only - 5 points] Write a Finite State Transducer for the K insertion spelling rule in English.

Q5. [10 points] The Soundex algorithm (Odell and Russell, 1922; Knuth, 1973) is a method commonly used in libraries and older Census records for representing people's names. It has the advantage that versions of the names that are slightly misspelled or otherwise modified (common in hand-written census records) will still have the same representation as correctly-spelled names. (e.g., Jurafsky, Jarofsky, Jarovsky, and Jarovski all map to J612). The rules are as follows:

a. Keep the first letter of the name, and drop all occurrences of non-initial a, e, h, i, o, u, w, y
b. Replace the remaining letters with the following numbers:
   b, f, p, v → 1
   c, g, j, k, q, s, x, z → 2
   d, t → 3
   l→4
   m, n → 5
   r→6
c. Replace any sequences of identical numbers, only if they derive from two or more letters that were *adjacent* in the original name, with a single number (i.e., 666 → 6).
d. Convert to the form Letter Digit Digit Digit by dropping digits past the third (if necessary) or padding with trailing zeros (if necessary).

Write a FST (with intermediate tapes if necessary) to implement the Soundex algorithm.

Q6. [5 points] Compute the minimum edit distance between `drive` and `brief`. Show all your work, including the table and the backtrace arrows.

Part II : Language Models

Language identification is the problem of taking a text in an unknown language and determining what language it is written in. N-gram models provide a very effective solution for this problem. For training, use the English (`EN.txt`), French (`FR.txt`), and German (`GR.txt`) texts made available in blackboard/piazza. For test, use the file `LangID.test.txt`. For each of the following problems, the output of your program has to contain a list of [line_id] [language] pairs:

ID	LANG
1	EN
2	FR
3	GR

Q7. [10 points] Implement a letter bigram model, which learns letter bigram probabilities from the training data. A separate bigram model has to be learned for each language (from each of the training files provided). Apply the models to determine the most likely language for each sentence in the test file (that is, determine the probability associated with each sentence in the test file, using each of the four language models).

Q8. [5 points] Implement a word bigram model, which learns word bigram probabilities from the training data. Again, a separate model will be learned for each language. Use Add-One smoothing to avoid zero-counts in the data. Apply the models to determine the language ID for each sentence in the test file.

Q9. [10 points] Implement a word bigram model, which learns word bigram probabilities from the training data. Again, a separate model will be learned for each language. Use Good Turing smoothing to avoid zero-counts in the data. Apply the models to determine the language ID for each sentence in the test file.

Q10. [required for Graduate students only 10 points] Implement a word Trigram model, which learns word trigram probabilities from the training data. Again, a separate model will be learned for each language. Use either Kneser-Ney smoothing or Katz Back-off (as described in chapter 4 of Jurafsky & Martin). Apply the models to determine the language ID for each sentence in the test file.

[5 points] Report the accuracy (#of correctly identified pairs/Total) for each language model above using the gold labels in `LangID.gold.txt`