LIN570: HW9 – hmm2 (100pts)

YOUR NAME (`UW NetID`)

Due date: 11pm on Dec 3, 2019 (Tuesday)

All the example files are under `~/dropbox/19-20/570/hw9/examples/`.

1. **Q1 (45 points):** Write a script, `viterbi.sh`, that implements the Viterbi algorithm. You can reuse some functions from your `check_hmm.sh` in Hw8.

   - The format is: `viterbi.sh input_hmm test_file  output_file`
   - Your code should work for any HMM, not just the HMM for ngram POS taggers:
     - Do not do anything special for BOS and EOS. Do not insert BOS marker or EOS marker to the input.
     - Use `input_hmm` as it is. Do NOT smooth it. For instance, if there is no transition probability line from state s_i to s_j, that means that it is impossible to go from s_i to s_j. If there is no emission line for state s_j and output symbol w_k, that means that s_j cannot generate w_k.
     - Your code should be able to handle unknown "word" in the observation: let the observation be "o_1 o_2 ... o_n". For each o_i, if o_i does not appear in the `input_hmm` at all, o_i is an *unknown* word and should be treated as a special output symbol `<unk>`. The special symbol can be generated by any state s_j whose probability $P($`<unk>` $\mid s_j)$ is larger than 0. If for some s_j, $P($`<unk>` $\mid s_j)$ is zero or does not appear in the input_hmm, that means s_j cannot generate this special symbol.
   - *input_hmm* is an input file:
     - `input_hmm` is a state-emission hmm and the output symbols are produced by the *to-states*. It has the same format as the HMM format in Hw8.
     - You can assume that the `input_hmm` does not contain any emission probability for empty string (i.e., a state cannot generate an empty string).
     - For Hw9, you don't need to check whether the three probability distributions (initial, transition, and emission ones) in the `input_hmm` satisfy the constraints that are checked by `check_hmm.sh` in Hw8. If a line contains a probability that is not in the $[0, 1]$ range, your code just prints out a warning message to `stderr` ("`warning: the prob is not in [0,1] range:  $line`", where `$line` is the line), ignore the line, and continue.
   - `test_file` is an input file;
     - Each line is an observation (i.e., a sequence of output symbols). For instance, if you use HMM for POS tagging, an observation will be a sentence (cf. `test.word`):

  - Once again, do not insert anything (e.g., BOS or EOS marker) to the observation.
- The format of the `output_file` (cf. `sys`) is `observ => state_seq lgprob`:
  - `state_seq` is the best state sequence for the observation. The length of `state_seq` should be equal to the length of `observ` plus one.
  - `lgprob` is $\log P(observ, state\_seq)$; $\log(x)$ is base-10 log.
  - If there is a tie (i.e., more than one state sequence with the highest probability), you can pick any of those sequences.

2. **Q2 (30 points):** Use `viterbi.sh` for *trigram* POS tagging:

   - The `input_hmm` for `viterbi.sh` is the one for a trigram POS tagger:
     - The state name has the format `tag1_tag2`, and the output symbol is produced by the *to-state*.
     - `hw9/examples/hmm[1-5]` are some examples of the input_hmm. For the transition and emission probability lines, please ignore anything after `##`.
   - Write your own script, `conv_format.sh`, to convert the format of the output file of `viterbi.sh`.
     - The format of the command line is `cat file1 | conv_format.sh > file2`
     - `file1` is the file created by `viterbi.sh`, and has the format `observ => state_seq lgprob`.
     - `file2` has the format `w1/t1 w2/t2 ... wn/tn`. where t_i is the second tag of the state that generates w_i.
     - For instance, if `file1` has a line "w_1 w_2 ... w_n => x_t_0 $t_0\_t_1$ $t_1\_t_2$ ... $t_{n-1}\_t_n$ lgprob", `conv_format.sh` should print "w_1/t_1 ... w_n/t_n" to `stdout`, which can then be redirected to `file2`. Note that x, t_0 (the two tags in the 1st state in the state sequence), and lgprob should NOT be included in the output string.
   - Run `calc_tagging_accuracy.pl` (which is given to you) to calculate the tagging accuracy.
     - The format is: `calc_tagging_accuracy.pl gold_standard sys_res > sys_res.acc`
     - `gold_standard` and `sys_res` have the format `w1/t1 w2/t2 ... wn/tn` (e.g., `test.word_pos`).
     - The gold standard for the file `test.word` is `test.word_pos`, and the `sys_res` is the file created by `conv_format.sh`
   - Fill out Tablel 1 with each of the HMM files under `hw9/examples/`. For instance, to get the accuracy for the first row in Table 1, you should run the following commands:

     ```
     viterbi.sh hmm1 test.word sys1
     cat sys1 | conv_format.sh > sys1_res
     calc_tagging_accuracy.pl test.word_pos sys1_res > sys1_res.acc 2>&1
     ```

   - Submit the files as specified in `submit-file-list`.

The submission should include:

- The `readme.`[`txt`|`pdf`] file that includes Table 1.
- `hw.tar.gz` that includes the files specified in `submit-file-list`).

Table 1: Tagging accuracy

HMM model	tagging accuracy
hmm1	
hmm2	
hmm3	
hmm4	
hmm5	