

LING 570: Hw7

Due on Nov 17

All the example files are under `dropbox/10-11/570/hw7/examples/`.

Q1 (60 points): Write a script, **viterbi.sh**, that implements the Viterbi algorithm. You should be able to use some functions from your `check_hmm.sh` in Hw6.

- The format is: `viterbi.sh input_hmm test_file output_file`
- The `input_hmm` is a state-emission hmm, which has the same format as the ones specified in Hw6.
- The format of the `test_file`: each line is an observation (i.e., a sequence of output symbols). For POS tagging, an observation will be a sentence (cf. **test.word**).
- The format of the `output_file` (cf. **sys**): “`observ => state_seq lgprob`”
`state_seq` is the best state sequence for the observation, and `lgprob` is $\lg P(\text{observ}, \text{state_seq})$.
- Note:
 - You can assume that the probabilities in the `input_hmm` have been smoothed already. 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 . And if there is no emission line for state s_j and output symbol w_k , that means that s_j cannot generate w_k . Do NOT try to smooth the probabilities in `input_hmm`.
 - Your code should be able to handle unknown “word” in the observation: let the observation be “ $o_1 o_2 \dots o_n$ ”. For each o_i , if o_i does not appear in the `input_hmm` at all, o_i is *unknown* and it can be generated by any state s_j with the probability $P(< unk > | s_j)$. You can assume that `input_hmm` includes emission probability for $P(< unk > | s_j)$ for every state s_j that can generate an unknown word.

Q2 (40 points): Build trigram models with **wsj_sec0.word_pos** as the training data and test the models on the test data **test.word**. It consists of several steps:

1. Run `create_3gram_hmm.sh` from Hw6 to create `hmm` from `wsj_sec0.word_pos` (use the `lambdas` specified in the table below, and `unk_prob_sec22` for $P(< unk > | tag)$).
2. Run `viterbi.sh` on `test.word` to produce an output file with the format “`observ => state_seq logprob`”.
3. Write a script, **conv_format.sh**, to convert the format of the output file of Step 2.
 - The command line is “`cat file1 | conv_format.sh > file2`”.
 - `file1` is the file created by Step 2, and `file2` has the format “`w1/t1 w2/t2 ... wn/tn`”.

4. Run `calc_tagging_accuracy.pl` 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**).
5. Fill out the following table.

For instance, to get the accuracy for the first row, you should run the following commands:

- `cat wsj_sec0.word_pos | create_3gram_hmm.sh hmm1 1.0 0 0 unk_prob_sec22`
- `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`

Table 1: Tagging accuracy

Expt Id	λ_1	λ_2	λ_3	tagging accuracy
1	1.0	0	0	
2	0.5	0.5	0	
3	0.2	0.8	0	
4	0.1	0.1	0.8	
5	0.2	0.3	0.5	

The submission should include:

- The hw7 note file that includes answers to Q2.
- The source and shell scripts in Q1 and Q2: **viterbi.sh**, **conv_format.sh**, and any scripts called by them.
- The files created in Q2: *hmm_i*, *sys_i*, *sys_i_res*, and *sys_i_res.acc*, where *i* is the experiment id in the first column of Table 1.