LING 570 – HW3

Q3 - Part (d)

What does the FST created by the trainer look like?

The FST created by the trainer is in the following format:

(S (S ("Token" "Tag" Probability)), where S = start and accept states

For examples:

(S (S "State" "NN" 0.0909090909090909))

(S (S "State" "NNP" 0.909090909090909))

How do your trainer and decoder work?

The trainer (trainer.sh / trainer.pl) accepts an input file consisting of tokens tagged with POS tags. The trainer program then separates the input content into individual pairs of token/tag, counts the occurrence of each token associated with each tag or tags and calculates the probability of each POS tag given the token that it is being associated to. Finally the calculated probability for each token/POS tag pair is used to generate a weighted Finite State Transducer (FST). The FST accepts token as input and outputs the likelihood of POS tag(s) given that token.

The decoder (decoder.sh) consists of two sub programs. The first program (format_input.pl) accepts an input file in text format and rewrites the text into a form that is expected by the FST, i.e enclosing each token with quotation marks (""). After the reformatted text is processed through the FST, the second program (decoder.pl) then post-processed the resultant FST output to generate the following final result per sentence:

w1/tag1 w2/tag2 … w_n/tag_n prob

What tagging accuracy do you get when running the commands in (c)?

The following results were obtained from running the commands in (c)

sent_num: counted_sent=1921 untagged=0 (0%), leng_mismatch=0 wrong_format=0
word_num: gold=46451, counted=46451, matched=44572

accuracy: overall=95.9548771824073%, among_counted=95.9548771824073%

untagged=0%

In (c), the test data used with decoder.sh is the same as the training data (once the tags are removed). What kind of problems will this unigram tagger encounter if the test data is not the same as the training data?

If the test data is not the same as the training data, it may result in tokens not being recognized for tagging. This would affect the whole sentence tagging. For example, if a token in a sentence is not recognized, the FST created from the training data will not tag the token and therefore causes the entire sentence to be rejected by the decoder.

Another issue is, the weighted FST created from the training data is based solely on the token/tag pairs that appear within the training data. Each token/tag pair and its conditional probability is context independent and may only be applicable to the context of the training data. If the tokens from the test data appear in a different context, then the probabilities of the tagged sentences output from the decoder may not necessarily yield the most accurate result.

Have you done something special for the quotation mark? If so, what's that and why is it necessary?

I have not done anything special to the quotation mark as my program seems to be able to process them without any issues. However, we need to ensure that the input test file is pre-process into a form that is accepted by the FST and carmel, i.e each token should be enclosed by quotation mark "token1" "token2" "tokenN", etc.

In addition, special treatment was also made to token/tag pair that appeared in this form:

=> summer\/winter/JJ.

Without additional treatment, the program would separate the token into "summer\" and "winter" based on the first "/". As a result, the trainer program would generate the following FST that would have produced an error when processed though the decoder:

(S (S "summer\" "winter" "JJ" 1))

The special treatment was to ensure that the trainer program recognizes the first "/" as part of the token, i.e "summer\/winter" and then correctly generates the following FST:

(S (S "summer\/winter" "JJ" 1))

End of HW3 – submitted by Wee Teck Tan

Student ID: 0937003

Course Name: LING 570