Project 2 Proposal

Gi Yoon Han, Rui Liang, Xiaoyun Quan

October 3, 2016

Kaggle team name: random_name

1 Preprocessing + baseline experiment

Before implementation, we preprocessed our data by the following steps:

- filter out the lines which do not have exactly 3 columns (these are the errors and end of paragraph tag i.e. empty line)
- replace all tags that doesn't contain "CUE" with "O" tag
- replace the first CUE-n tag with B-CUE, $n = 1, 2, 3, ...$
- replace the rest of CUE-n tag with I-CUE, $n = 1, 2, 3, ...$

Now that the data has been 8i formatted, we can run an experiment with baseline system. Our baseline is a dictionary consisting of words tagged with `B-CUE` or `I-CUE` from the first 1% of the entire training data. We will tag any word in the test file whenever it appears in the dictionary.

To run our code, please unzip the data file, and put the folder `nlp_project2_uncertainty` in the same location as the script `training.py`, then run `training.py` and you should get the baseline output on screen and in `csv`.

The precision (and same as recall here) metric turns out to be 0.45742 and 0.00716 for predictions on sentence-wise and phrase-wise respectively. This suggests that more reliable methods are needed for better precision.

2 Experiment design

Considering that Hidden Markov Models (HMM) is probabilistic generative, we decided to implement it as our first attempt. We will start with calculate transition and lexical generation probabilities from the training data, followed by running the viterbi algorithm available from library. Note that only the first and third columns are involved here. To sum up, we want to maximize

$$P(t_1, ..., t_n | w_1, ..., w_n) = \frac{P(t_1, ..., t_n) \cdot P(w_1, ..., w_n | t_1, ..., t_n)}{P(w_1, ..., w_n)}$$

where t_i are the predicted tag for the i-th token in the test text, and w_i the observed tag corresponding to the i-th token in observation. We are making two assumptions here to approximate the right-hand side:

- assume that a word appears in a category independent of its neighbors so that $P(w_1, ..., w_n | t_1, ..., t_n) \approx \prod_{i=1,..,n} P(w_i | t_i)$
- assume bigram model so that $P(t_1, ..., t_n) \approx \prod_{i=1,..,n} P(t_i | t_{i-1})$

3 Extension

We are planning to do two extensions:

- implement CRF (), since CRF is more relaxing in the sense that less assumptions are made and it's flexible with definining features so that we can assign different weights to different situations accordingly;
- try BILOU tagging to see if there will be any difference in performance than BIO tagging.