Writeup Zhang Ningyi

1. Language features

    1) Examples with word “de” or “De”. Because Dutch frequently uses “de”, which seldom exists in English sentences.
    2) Examples with word “a” or “an”. Because English frequently uses “a” or “an”, which seldom exists in Dutch sentences.
    3) Examples with word ended with “ed”. Because English frequently uses “ed” as form of past tense, which seldom exist in Dutch sentences.
    4) Examples with word contains “aa”. Because Dutch has many words containing “aa”, which seldom exist in English words.
    5) Examples with word “the” or “The”. Because English frequently uses “the”, which seldom exist in Dutch sentences.

2. Decision tree

    After reading in examples, convert each example with Boolean of the 5 above feature, iteratively build the decision tree:
    In each step use the method of information gain to determine the next attribute, then group the attributed examples to be the left and right child of the current node. Then process the same algorithm in the two children.
    In the process, once an attribute has only one kind of language, this node then stop further iteration. Language that has larger portion in one group will be regard as the predict of the group.
    In prediction process, the input example will be converted to Boolean of the 5 features. Then from the root of the decision tree, move the position according to the Boolean of the feature. After getting to the final position when all 5 features are examined / arrive the max depth of the decision tree / current node has no child, the prediction of the landing node will be the prediction of this input example. If it agrees with the example’s label, this prediction is correct.
    In my own testing, the error rate will exist when max depth is about 3, rather than 5. It happens probably because some boundary examples or examples that is unhandled within the 5 features will possibly gather or gain more weight when doing deep attribution, thus disturb the prediction.

    branch ___T_ : total: 18 attribute: 3 which is true en: 0 nl: 18 informain gain: 0.0 predict: nl
    branch ___F_ : total: 27 attribute: 3 which is false en: 23 nl: 4 informain gain: 0.27594386796016235 predict: en
    branch _T_F_ : total: 11 attribute: 1 which is true en: 11 nl: 0 informain gain: 0.0 predict: en

branch _F_F_ : total: 16 attribute: 1 which is false en: 12 nl: 4 informain gain: 0.46666279373280395 predict: en
branch _FTF_ : total: 7 attribute: 2 which is true en: 7 nl: 0 informain gain: 0.0 predict: en
branch _FFF_ : total: 9 attribute: 2 which is false en: 5 nl: 4 informain gain: 0.5900048960119098 predict: en
branch _FFFT : total: 4 attribute: 4 which is true en: 4 nl: 0 informain gain: 0.0 predict: en
branch _FFFF : total: 5 attribute: 4 which is false en: 1 nl: 4 informain gain: 0.2916919971380596 predict: nl
branch TFFFF : total: 2 attribute: 0 which is true en: 0 nl: 2 informain gain: 0.0 predict: nl
branch FFFFF : total: 3 attribute: 0 which is false en: 1 nl: 2 informain gain: 0.0 predict: nl

test of max depth 5

3. After reading in examples, convert each example with Boolean of the 5 above feature, set initial weight of example evenly. Then iteratively choose attribute out of the 5 features to boost:
   The attribute with the lowest weight of error examples will be the next hypothesis. Examine the examples using this hypothesis. Each time an example has different language label from the prediction, increase the error of current hypothesis by the weight of the example. Each time the label agrees with the prediction, decrease the weight of the example. After each iteration, update weight of the current hypothesis. In my own testing, tree 3 and 4 is mostly used, while tree 1 is least frequently used as hypothesis. When number of boost larger than 3, the error rate of the hypothesis increased, and the performance of prediction begin to drop. Probably it is because there are examples that all features cannot cover, so that its weight will keep increasing in each iteration.

   0th boost, hypotheses: attribute 3, error rate: 0.17777777777777778 weight: 1.5314763709643886
   1th boost, hypotheses: attribute 0, error rate: 0.3560224089635854 weight: 0.5926702535737276
   2th boost, hypotheses: attribute 4, error rate: 0.2303018079342936 weight: 1.2066078225156178
   3th boost, hypotheses: attribute 2, error rate: 0.32253285143006344 weight: 0.7421560664351793
   4th boost, hypotheses: attribute 3, error rate: 0.43250688682440397 weight: 0.2716303643699452

   Test of boosting 5 stumps