CSCI 6350/7350 Notes on text classification using Naïve Bayes

1. Build a Naïve Bayes classifier to determine whether a user would be interested or not interested in a document. α=1.

   Assume the use has indicated he is interested in the following documents :
   Doc 1: victory garden
   Doc 2: square foot organic gardening
   Doc 3: the victory garden companion
   Doc 4: victory gardens for organic foods

   And he is not interested in the
   Doc 1: four season harvest
   Doc 2: gardening bible
   Doc 3: victory gardening art poster
   Doc 4: victory gardening recipe

   Show whether the classifier will determine the following document to be of interest or not of interest:
   Grow organic food with victory garden method

2. Given the documents below that belong to 3 different categories: Artificial Intelligence(AI), Parallel Processing(PP), and Software Engineering(SE),

   (1) construct a naïve Bayes classifier. α=1.

   AI:
   Doc 1: new reinforcement learning method
   Doc 2: machine learning is one important AI method

   PP:
   Doc 1: load balancing is important
   Doc 2: balancing the documents on systems

   SE:
   Doc 1: System verification method

   (2) Classify the following two documents into the correct category using the naïve Bayes classifier built.

   Test doc: heuristic search based reinforcement learning method

TRAINMULTINOMIALNB($\mathbb{C}, \mathbb{D}$)

1 $V \leftarrow$ EXTRACTVOCABULARY($\mathbb{D}$)
2 $N \leftarrow$ COUNTDOCS($\mathbb{D}$)
3 **for each** $c \in \mathbb{C}$
4 **do** $N_c \leftarrow$ COUNTDOCSINCLASS($\mathbb{D}, c$)
5 $prior[c] \leftarrow N_c/N$
6 $text_c \leftarrow$ CONCATENATETEXTOFALLDOCSINCLASS($\mathbb{D}, c$)
7 **for each** $t \in V$
8 **do** $T_{ct} \leftarrow$ COUNTTOKENSOFTERM($text_c, t$)
9 **for each** $t \in V$
10 **do** $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)}$
11 **return** $V, prior, condprob$

APPLYMULTINOMIALNB($\mathbb{C}, V, prior, condprob, d$)

1 $W \leftarrow$ EXTRACTTOKENSFROMDOC(V, d)
2 **for each** $c \in \mathbb{C}$
3 **do** $score[c] \leftarrow \log prior[c]$
4 **for each** $t \in W$
5 **do** $score[c] \mathrel{+}= \log condprob[t][c]$
6 **return** $\arg\max_{c \in \mathbb{C}} score[c]$

```
TRAINBERNOULLINB(C, D)
1  V ← EXTRACTVOCABULARY(D)
2  N ← COUNTDOCS(D)
3  for each c ∈ C
4  do N_c ← COUNTDOCSINCLASS(D, c)
5     prior[c] ← N_c / N
6     for each t ∈ V
7     do N_ct ← COUNTDOCSINCLASSCONTAININGTERM(D, c, t)
8        condprob[t][c] ← (N_ct + 1) / (N_c + 2)
9  return V, prior, condprob

APPLYBERNOULLINB(C, V, prior, condprob, d)
1  V_d ← EXTRACTTERMSFROMDOC(V, d)
2  for each c ∈ C
3  do score[c] ← log prior[c]
4     for each t ∈ V
5     do if t ∈ V_d
6           then score[c] += log condprob[t][c]
7           else score[c] += log(1 − condprob[t][c])
8  return arg max_{c∈C} score[c]
```