ReadMe: this part is for replication needs. Please contact me for the zip-file.

To replicate Running, please unzip and run WindowsRun.bat(Windows). Or LinuxRun.sh(UNIX)

The result will be generated as root/result/[question#]/part-00000, use notepad to check.

environment and exe folders includes all necessary environment, and dataset, please do not remove. Scala source code can be found under sourceCode folder, where ShareSparkMethod is a helper class.

LinxiaoBai HW2

QuestionA:

rahj myuw wkrx nfmq geeb eoap ezsd zspm qcxj tgdy
xkrp vmwm mpmp ylwr kvme ozgb oqay hufo jcmx ghpt
eqrg fnzd jsjg gwxh tnsk cwaj iwsn lzxi fflo prez
etuk wwht wxiv errc itac uefy jktx afwa xbdx lixx
dwxh jcts alrm gbcu spzi iavk adex abdi wxhq jbhg
yzdc cbex yped ctcm vzum etfs aieg juep kgox xaui
dbyf lwyc ilhp taoo bmpl degy rtkt cflq wlxg jpih
cjar szbn uojs imcb olzh eqrq jlbq orre pnhu agxq
ygan bbpt jtiz bjdf muky xxbm ilea ksrb javl dthi
tldf owir hhru atgs hxbx tidy jofg pqrn abgd knit

3known words

QuestionB

iieo leto aewe atoi etdo ihmk saoe tuie wkil eriw
wepo ktim eost tloi ekte slto tlkr hatr aoey dean
eunr itys ryot aodd iair iitm iitt nlme enem wiwl
etdl saed eusu vain oiyl npai dnui mrin ocys aepo
wttu dlel iaeh eony ntam duoo atos oadd owno eyyd
okyr ioao nmtt mook lnst ifwo yuna wneh eaee saga
arae iirn ulma tiet ydym hlde soih nafl ymna lmee
sdwe oyet niui ywrm nmye elkt llys mnoo bnoa natn
omor etda yosh daem irud ehan tean lrii ddak emne
lyya clae inta lyiy nuiu uouw enrl enti atsk freo

2known word

QuestionC

noth spam auli iant yoth then llit ithe ilen them
erun inom aino doem ithe cono erea wami like iath
lldo eren thev nota nyor scou itsp nome liat ilam
itce yoma huli cewi buly them eram yots anou doul

dour them bspa ldot adlo eree wami rema noul ldou
noth amia heyw ispa nouy teme iake aian iado yota
idot doth wito ikee wits noul noul noul wonc ewou
hema lili nche oule scha amyw hema yoth agar scon
yots fren yoee heta fret bspa ithe ldot amak wats
ldon herk itre keno fith inob from coth ilde sewh

13known words

QuestionD

itso aill frea noul ayou ldom noth spam ithe spam
adoo illy hant ithe itsp noul yoro emia lisp ldea
dewd that ikew iflo iami spam buld ithe iaga nots
lith rema ilik dobs tche noth spam wome ldea iker
frud anot ikew ithe ildo your honc noul ikea youl
from spam ithe park iket othe noul ouyw reth spid
hill iken notc frke illy sots wdet ikne sere erke
athi yooz ikee seen woul spam lits whem thee doth
inot your tsay aiam urea itse noul inot ithe cewd
woul afre tspa ikee ilik noul ally your spam udou

20known words

QuestionEA

rahj myuw wkrx nfmq geeb eoap ezsd zspm qcxj tgdy
xkrp vmwm mpmp ylwr kvme ozgb oqay hufo jcmx ghpt
eqrg fnzd jsjg gwxh tnsk cwaj iwsn lzxi fflo prez
etuk wwht wxiv errc itac uefy jktx afwa xbdx lixx
dwxh jcts alrm gbcu spzi iavk adex abdi wxhq jbhg
yzdc cbex yped ctcm vzum etfs aieg juep kgox xaui
dbyf lwyc ilhp taoo bmpl degy rtkt cflq wlxg jpih
cjar szbn uojs imcb olzh eqrq jlbq orre pnhu agxq
ygan bbpt jtiz bjdf muky xxbm ilea ksrb javl dthi
tldf owir hhru atgs hxbx tidy jofg pqrn abgd knit
3 known words

Question EB

tedu nree wtyu odre dead toor tkos nsew ooie ophn
trea hhtm olor ioet pese eloo ywoh aaun efdo doet
ekei wmeh aooe aben rwee uiwd etcl ensy aoea oruh
osea pihn lino void edfh dwas acot vsle lfhy wrrr
fyts hrti diot rfri nlsy dlee snio sooa srwm rrse
orri nshb kgns yepi hapt eorh chyo rwth yeeb larc
hgib hehh olfi aeoo valf dtce noyl eogt hcin poos
baet rlpw odte hinu mhrw erai htcc pdco holb eiim

asou etfs emtp egis fhit hcen deht vyea eres lgra
ogae shcm aata tget atso tdsr eket orua dldc enei

4known words

Question EC

oean herk adof acam usto atie gedo sctl ofam woha
evee wler rayh inno pema altl byou fane asth tiap
wiss thof shes mouc wame sech spin wein ingr ilym
wlyo sosi ofom gern peru chin whas whep isof dsse
heap wham cofi veme otot inth cany cour olde ween
inti icor foul wodo yero inde woll wers nshe falf
cono tlol thei onin sppi stha cera thes itly they
when sudu mefa grat thar hend aide asce dess thar
toth tong bane sing ilyo hild itit whem bono idit
spom tron lved emed fain nlev ouen sing ossh pldi
15 known words

Question ED

sall ithe ueti cany pica pens ndea cort cion thap
town whad beek ctoo shav site nsic belf wnge lyse
heak okin herc agru stha hard whey heas syth tofi
inds hedu ledg inge amag thei cone sund bing fast
dovo llid ttte mess vini ilum wast hess thin ondo
sime amus ndow whad hest sirs sthe mere tess that
ther olyd stra athe inag hofi bers helo apro alie
ssto ngss tory tome ther whto hare told wion serc
ceft bedu athe bund toss tous ting heca ours tews
atac ilfe sess bula ssts when wher body waus ined

24 known words

Question F

From the 10*10 word table we see that, as the model get more and more complicated and depend on previous observations, the more legit the result become.

As pure random model and random model based on letter frequency only collects 3-4 know words by chance, and the conditional models generates 15 words at least.

Conditional model based 2 previous observations generates even more than the markov model
We can also conclude that as model become more concrete, the generated words become replicated more often. As “spam”, “your” shows multiple times in one simulation.

Also, an other observation is that model trained out of saki_story.txt yields better result than spamiam.txt in general. This is determined by the nature of the datasets

Code: in Scala (using spark engine) if running required, I am happy to provide the source code and environment for replication.

```
package ECE443HW2
import Tools.{ShareSparkMethod => SSM}
import org.apache.spark.{ SparkContext}

import scala.collection.mutable.ArrayBuffer
import scala.util.Random


/**
* Created by Linxiao Bai on 9/22/16.
*/
object HW2{

 def weightedRandom(weight:Map[Char,Int]):Char={
  val totalWeight=weight.reduce((x,y)=>(x._1,x._2+y._2))._2
  var random= Random.nextInt(totalWeight)
  for(i <- weight.keySet){
   if(random-weight(i)<0)
    return i
   else random=random-weight(i)
  }
  return 's'
 }


 def weightedRandom2(weight:Map[Char,Map[Char,Int]],inChar:Char,charMap:Map[Char,Int]):Char={
  if(!weight.contains(inChar)) return weightedRandom(charMap)
  else{
   val curMap=weight(inChar)
   return weightedRandom(curMap)
  }
 }

 def
weightedRandom3(weight:Map[String,Map[Char,Int]],inString:String,charMap2:Map[Char,Map[Char,Int]],char
Map:Map[Char,Int]):Char={
  if(!weight.contains(inString)) return weightedRandom2(charMap2,inString(1),charMap)
  else{
   val curMap=weight(inString)
   return weightedRandom(curMap)
  }
 }
```

```
def question1(args:Array[String],sc:SparkContext) ={

  val alphabet="ABCDEFGHIJKLMNOPQRSTUVWXYZ".toLowerCase()
  val r=new Random()
     r.setSeed(1L)
  val ab=new ArrayBuffer[ArrayBuffer[String]]()

  var i=0

  while (i<10){
   val elem=new ArrayBuffer[String]()
   var j=0

   while (j<10){
    var k=0
    j+=1
    var word=""
    while(k<4){
     val indx=r.nextInt(alphabet.length())
     word=word+alphabet(indx)
     k+=1
    }
    elem+=word
   }
   ab+=elem
   i+=1
  }

  SSM.delPath(args(0))
  sc.parallelize(ab)
   .map(x => x.mkString(" "))
   .saveAsTextFile(args(0))




}


def question2(args:Array[String],sc:SparkContext)= {
  val charMap = sc.textFile(args(1)) //This step efficiently get each char and their count as a (char,#) map
   .filter(x => x != "")
   .map(x => x.replaceAll("[^A-Za-z0-9]", "").toLowerCase())
   .flatMap(x => x.toCharArray)
   .map(x => (x, 1))
   .reduceByKey(_ + _)
   .collect
   .toMap
```

```
  val ab=new ArrayBuffer[ArrayBuffer[String]]()
  ab.clear()
  var i = 0

  while (i < 10) {
    val elem = new ArrayBuffer[String]()
    var j = 0

    while (j < 10) {
      var k = 0
      j += 1
      var word = ""
      while (k < 4) {

        //this part is different from question1, representing a different way of selection
        word = word + weightedRandom(charMap)
        k += 1
      }
      elem += word
    }

    ab += elem
    i += 1
  }

  SSM.delPath(args(0))
  sc
    .parallelize(ab)
    .map(x => x.mkString(" "))
    .saveAsTextFile(args(0))
}


def question3(args:Array[String],sc:SparkContext) ={
  val conditionMap=sc
    .textFile(args(1))
    .map(x=>x.split(" "))
    .map{x=>
      x.map(x=>x.replaceAll("[^A-Za-z0-9]", "").toLowerCase())
    }
    .flatMap(x=>x)
    .map{x=>
      val ab=new ArrayBuffer[String]()
      var last='*'
      var next='*'
      val it=x.iterator
      while(it.hasNext){
        last=next
```

```
      next=it.next()
      ab+=new StringBuilder().append(last).append(next).toString()
    }
    ab
  }
  .flatMap(x=>x)
  .map(x=>(x(0),x(1).toString))
  .reduceByKey{(x,y)=>
    x+y
  }
  .collect()
  .map{x=>
  val charA =sc.parallelize(x._2.toCharArray)
    .map(x=>(x,1))
    .reduceByKey(_+_)
    .collect()
    .toMap
  (x._1,charA)
}
.toMap

//    Doing simulation
val charMap = sc.textFile(args(1)) //This step efficiently get each char and their count as a (char,#) map
  .filter(x => x != "")
  .map(x => x.replaceAll("[^A-Za-z0-9]", "").toLowerCase())
  .flatMap(x => x.toCharArray)
  .map(x => (x, 1))
  .reduceByKey(_ + _)
  .collect
  .toMap


val ab=new ArrayBuffer[ArrayBuffer[String]]()
var i=0
while (i<10){
  val elem=new ArrayBuffer[String]()
  var j=0
  while (j<10){
    var k=0
    j+=1
    var lastletter='*'
    var word=""
    while(k<4){
      //this part is different from question1, representing a different way of selection
      //draw first word based on the weight of the first elem
      lastletter=weightedRandom2(conditionMap,lastletter,charMap)
      word=word+lastletter
      k+=1
    }
```

```
      elem+=word
    }

    ab+=elem
    i+=1
  }

  SSM.delPath(args(0))
  sc.parallelize(ab)
    .map(x => x.mkString(" "))
    .saveAsTextFile(args(0))


}



def question4(args:Array[String],sc:SparkContext)={
  val charMap = sc.textFile(args(1)) //This step efficiently get each char and their count as a (char,#) map
    .filter(x => x != "")
    .map(x => x.replaceAll("[^A-Za-z0-9]", "").toLowerCase())
    .flatMap(x => x.toCharArray)
    .map(x => (x, 1))
    .reduceByKey(_ + _)
    .collect
    .toMap

  val conditionMap=sc
    .textFile(args(1))
    .map(x=>x.split(" "))
    .map{x=>
      x.map(x=>x.replaceAll("[^A-Za-z0-9]", "").toLowerCase())
    }
    .flatMap(x=>x)
    .map{x=>
      val ab=new ArrayBuffer[String]()
      var last='*'
      var next='*'
      val it=x.iterator
      while(it.hasNext){
        last=next
        next=it.next()
        ab+=new StringBuilder().append(last).append(next).toString()
      }
      ab
    }
    .flatMap(x=>x)
    .map(x=>(x(0),x(1).toString))
    .reduceByKey{(x,y)=>
```

```
  x+y
 }
 .collect()
 .map{x=>
  val charA =sc.parallelize(x._2.toCharArray)
   .map(x=>(x,1))
   .reduceByKey(_+_)
   .collect()
   .toMap
  (x._1,charA)
 }
 .toMap


val conditionMap2=sc.textFile(args(1))
 .flatMap(x=>x.split(" "))
 .map(x => x.replaceAll("[^A-Za-z0-9]", "").toLowerCase())
 .filter(x=>x.length>2)
 .map{x=>

  val ab=new ArrayBuffer[String]()
  val it=x.iterator
  var one='*'
  var two='*'
  var three='*'
  while(it.hasNext){
   one=two
   two=three
   three=it.next()
   ab+= new StringBuilder().append(one).append(two).append(three).toString()
  }
  ab
 }
 .flatMap(x=>x)
 .map(x=>(x(0).toString+x(1).toString,x(2).toString))
 .reduceByKey{(x,y)=>
  x+y
 }
 .collect()
 .map{x=>
  val charA =sc.parallelize(x._2.toCharArray,1)
   .map(x=>(x,1))
   .reduceByKey(_+_)
   .collect()
   .toMap
  (x._1,charA)
 }
 .toMap
```

```
   //Simulator here
  val ab=new ArrayBuffer[ArrayBuffer[String]]()
  var i=0
  while (i<10){
   val elem=new ArrayBuffer[String]()
   var j=0
   while (j<10){
    var k=0
    j+=1
    var lastletter='*'
    var word=""
    while(k<4){
     //this part is different from question1, representing a different way of selection
     //draw first word based on the weight of the first elem
     //draw second element based on the method of question3

     if(word.length<3) {
      lastletter = weightedRandom2(conditionMap, lastletter, charMap)
     }
     else{
      lastletter=weightedRandom3(conditionMap2,word.substring(word.length-
2,word.length),conditionMap,charMap)
     }
     word = word + lastletter
     k+=1
    }
    elem+=word
   }

   ab+=elem
   i+=1
  }

  SSM.delPath(args(0))
  sc.parallelize(ab)
   .map(x => x.mkString(" "))
   .saveAsTextFile(args(0))
 }


 def question5(args:Array[String],sc:SparkContext): Unit ={
  val intext=args(0)
  question1(Array(args(1),intext),sc)
  question2(Array(args(2),intext),sc)
  question3(Array(args(3),intext),sc)
  question4(Array(args(4),intext),sc)
 }
```

```
  def main(args: Array[String]) {
    if(args.length<1){
      System.err.println("args#err")
      System.exit(1)
    }

    val sc=SSM.getSparkContextLocal("ECEHW2")

    question1(Array(args(1)),sc)
    question2(Array(args(2),args(0)),sc)
    question3(Array(args(3),args(0)),sc)
    question4(Array(args(4),args(0)),sc)
    question5(Array(args(5),args(6),args(7),args(8),args(9)),sc)

    println("Run Finished")
  }

}
```