what would i say?

http://what-would-i-say.com/

Try it!!

Markov

- The infinite monkey theorem
Markov

- Training text
  - ex. Huckleberry Finn
- Build a map from text
  - ‘e’ is followed by ‘a’ 30% of the time
  - ‘a’ is followed by ‘t’ 20% of the time
- Generate random text

Markov

- \texttt{bbbababbbababa}  
  3-gram

- \begin{tabular}{|c|c|c|}
  \hline
  bbb & & \\
  & & \\
  & & \\
  & & \\
  \hline
\end{tabular}
Markov

• bbbabbbabbbbaba

3-gram

<table>
<thead>
<tr>
<th>bbb</th>
<th>bba</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• bbbabbbabbbbaba

3-gram

<table>
<thead>
<tr>
<th>bbb</th>
<th>bba</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Markov

• bbbababbababa

3-gram

<table>
<thead>
<tr>
<th>bbb</th>
<th>bba</th>
</tr>
</thead>
<tbody>
<tr>
<td>bba</td>
<td>bab</td>
</tr>
<tr>
<td>bab</td>
<td>abb</td>
</tr>
</tbody>
</table>

Markov

• bbbabababbaba

3-gram

<table>
<thead>
<tr>
<th>bbb</th>
<th>bba</th>
</tr>
</thead>
<tbody>
<tr>
<td>bba</td>
<td>bab</td>
</tr>
<tr>
<td>bab</td>
<td>abb</td>
</tr>
<tr>
<td>abb</td>
<td>bba</td>
</tr>
</tbody>
</table>
Markov

- `bbbabbabbbababa` 3-gram

<table>
<thead>
<tr>
<th>3-grams</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>bbb</td>
<td>bba</td>
</tr>
<tr>
<td>bba</td>
<td>bab</td>
</tr>
<tr>
<td>bab</td>
<td>abb</td>
</tr>
<tr>
<td>abb</td>
<td>bba</td>
</tr>
</tbody>
</table>

Markov

- `bbbabbabbbbababa` 3-gram

<table>
<thead>
<tr>
<th>3-grams</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>bbb</td>
<td>bba</td>
</tr>
<tr>
<td>bba</td>
<td>bab</td>
</tr>
<tr>
<td>bab</td>
<td>abb</td>
</tr>
<tr>
<td>abb</td>
<td>bba</td>
</tr>
<tr>
<td></td>
<td>bbb</td>
</tr>
</tbody>
</table>
Markov

• bbbabbabbbbaba

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bbb</td>
<td>bba</td>
<td>bbb</td>
<td>bba</td>
</tr>
<tr>
<td>bba</td>
<td>bab</td>
<td>bab</td>
<td>bab</td>
</tr>
<tr>
<td>bab</td>
<td>abb</td>
<td>abb</td>
<td>aba</td>
</tr>
<tr>
<td>abb</td>
<td>bba</td>
<td>bbb</td>
<td></td>
</tr>
<tr>
<td>aba</td>
<td>bab</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3-gram

• “bbb” followed by “bba” 66% and “bbb” 33%
• “bba” always followed by “bab”
Markov

- Training text
  - ex. Huckleberry Finn
- Build a map from text
  - ‘e’ is followed by ‘a’ 30% of the time
  - ‘a’ is followed by ‘t’ 20% of the time
- Generate random text

Markov

- Recitation 6 – Help with Markov
  - Posted online, you may complete it BEFORE recitation
  - Get help from TAs and peers during recitation
Markov

- Step 1: Build your map
  - Make a simple text file (like the one we just used)
  - Print your map to make sure it is correct
- Step 2: Generate text from your map
  - Your text must be IDENTICAL to the bruteForce method

Markov

- “You’ll implement a class named MapMarkovModel that extends the abstract class AbstractModel.”

- What does that mean?
Today

- extends
- Object
- abstract
- interface

with robots

Today

- extends
- Object
- abstract
- interface

with robots
extends

class BendingRobot
doBending()
useElectricity()

class ClampingRobot
doClamping()
useElectricity()
extends

class BendingRobot
doBending()
useElectricity()

class ClampingRobot
doClamping()
useElectricity()

class EvilSantaRobot
doPunishNaughtyChildren()
useElectricity()
extends

class Robot{
    public void useElectricity{

    }

    // other common robot functions

}

extends

class BendingRobot extends Robot{

    doBending(){
        useElectricity();
    }

}

- BendingRobot inherits useElectricity from Robot.
extends

class GoldBendingRobot extends BendingRobot{
    doSuperBending();
    doBending();
    useElectricity();
}

extends

- BendingRobot – Superclass
  - GoldBendingRobot – Subclass

  Subclass inherits from superclass

GoldBendingRobot goldBot = new GoldBendingRobot();
goldBot.doBending();
goldBot.doSuperBending();
extends

• BendingRobot – Superclass
  • GoldBendingRobot – Subclass

• Superclass does not inherit from subclass

BendingRobot someBender = new GoldBendingRobot();
someBender.doBending();
someBender.doSuperBending(); //NOT ALLOWED

extends

• A extends B – A inherits all functions and variables from B
• Subclass A can be used any place superclass B can.
• Subclass A can “override” functions in superclass B
• Always use the most general type possible
Today

- extends
- Object
- abstract
- interface

with robots

Object

- Object (Java class) – superclass of your class
- All objects inherit
  - .equals()
  - .toString()
  - .hashCode()

- You can Override superclass with your own code
Today

- extends
- Object
- abstract
- interface

Markov

- “You’ll implement a class named MapMarkovModel that extends the abstract class AbstractModel.”

- What does that mean?
abstract

class BendingRobot
doBending()
useElectricity()

class ClampingRobot
doClamping()
useElectricity()

class EvilSantaRobot
doPunishNaughtyChildren()
useElectricity()

abstract

abstract class GenericRobot{
    abstract public void useElectricity();
    //all robots use electricity
    //but it may be different!!!!

    public void beep(){
        System.out.println("Beep!");
        //this is the same for all robots!!!!
    }
}

2/19/14
abstract

class BendingRobot extends GenericRobot {
  //must implement abstract methods
  public void useElectricity() {
    //your code goes here
  }
}

abstract

//doesn't matter what kind of Robot
GenericRobot myRobot = new BendingRobot();
BendingRobot bendRobot = new BendingRobot();
myRobot.beep();
myRobot.useElectricity();
GenericRobot otherBot = new GenericRobot(); //NOT ALLOWED
abstract

- abstract superclass contains abstract functions
- Subclass A of abstract superclass B must implement the abstract functions
- Can have variables of abstract superclass type, but cannot create objects of abstract superclass (can never use new)

Today

- extends
- Object
- abstract
- interface

with robots
abstract class Robot{
  
  abstract public void useElectricity();
  //all robots use electricity
  //but it may be different!!!!
  
  public void beep(){
    System.out.println("Beep!");
    //this is the same for all robots!!!!
  }
}

interface MoveRobot{
  
  void moveForward();
}

*all of our methods are abstract
interface
class BendingRobot implements MoveRobot{
    doBending(){}
    moveForward(){}
}

• BendingRobot must implement methods from MoveRobot.

interface
class BendingRobot extends Robot implements MoveRobot{
    doBending(){}
    moveForward(){}
    useElectricity(){}
}

• BendingRobot must implement methods from MoveRobot, but inherits methods from Robot.
interface

• Similar to a superclass, but has no implemented methods
• You **implement** an interface in the same way you **extend** a superclass
• You can implement many interfaces, but only extend one superclass

Today

• extends
• Object
• abstract
• interface

with robots