Word N-Grams

Order-One Concepts

Duke
UNIVERSITY

MarkovWordOne Class

- Use Concepts from Markov Programs
  - Usable, tested interface: IMarkovModel
  - Client programs continue to work
- Implementation changes; interface doesn't
  - Abstraction in software at its best!
- String **`myText`** to String[] **`myText`**
  - Searching for words rather than characters
  - Need to create new helper methods

Initialize MarkovWordOne

- Instance variables, constructor

```
public class MarkovWordOne implements IMarkovModel {

  private String[] myText;
  private Random myRandom;

  public MarkovWordOne() {
    myRandom = new Random();
  }

  public void setTraining(String text){
    myText = text.split("\\s+");
  }
```

Initialize MarkovWordOne

- Instance variables, constructor

```
public class MarkovWordOne implements IMarkovModel {

  private String[] myText;
  private Random myRandom;

  public MarkovWordOne() {
    myRandom = new Random();
  }

  public void setTraining(String text){
    myText = text.split("\\s+");
  }
```

Initialize MarkovWordOne

- Instance variables, constructor

```
public class MarkovWordOne implements IMarkovModel {

  private String[] myText;
  private Random myRandom;

  public MarkovWordOne() {
    myRandom = new Random();
  }

  public void setTraining(String text){
    myText = text.split("\\s+");
  }
```

Initialize MarkovWordOne

- Instance variables, constructor
  - Also **`setTraining`** method, create array

```
public class MarkovWordOne implements IMarkovModel {

  private String[] myText;
  private Random myRandom;

  public MarkovWordOne() {
    myRandom = new Random();
  }

  public void setTraining(String text){
    myText = text.split("\\s+");
  }
```

MarkovWordOne getRandomText

- Interface specifies method to return text

```
public String getRandomText(int numWords){
    StringBuilder sb = new StringBuilder();
    int index = myRandom.nextInt(myText.length-1);
    String key = myText[index];
    sb.append(key);
    sb.append(" ");
    // for loop not shown
    return sb.toString().trim();
}
```

MarkovWordOne getRandomText

- Interface specifies method to return text
  - String **`getRandomText(int numChars)`**

```
public String getRandomText(int numWords){
    StringBuilder sb = new StringBuilder();
    int index = myRandom.nextInt(myText.length-1);
    String key = myText[index];
    sb.append(key);
    sb.append(" ");
    // for loop not shown
    return sb.toString().trim();
}
```

MarkovWordOne getRandomText

- Interface specifies method to return text
  - String **`getRandomText(int numChars)`**
  - Type of parameter is meaningful, not name!

```
public String getRandomText(int numWords){
    StringBuilder sb = new StringBuilder();
    int index = myRandom.nextInt(myText.length-1);
    String key = myText[index];
    sb.append(key);
    sb.append(" ");
    // for loop not shown
    return sb.toString().trim();
}
```

MarkovWordOne getRandomText

- Interface specifies method to return text
  - String **`getRandomText(int numChars)`**
  - Type of parameter is meaningful, not name!
  - Add words not characters, manage spaces

```
public String getRandomText(int numWords){
    StringBuilder sb = new StringBuilder();
    int index = myRandom.nextInt(myText.length-1);
    String key = myText[index];
    sb.append(key);
    sb.append(" ");
    // for loop not shown
    return sb.toString().trim();
}
```

MarkovWordOne getRandomText

- Interface specifies method to return text
  - String **`getRandomText(int numChars)`**
  - Type of parameter is meaningful, not name!
  - Add words not characters, manage spaces

```
public String getRandomText(int numWords){
    StringBuilder sb = new StringBuilder();
    int index = myRandom.nextInt(myText.length-1);
    String key = myText[index];
    sb.append(key);
    sb.append(" ");
    // for loop not shown
    return sb.toString().trim();
}
```

MarkovWordOne getRandomText

- Interface specifies method to return text
  - String **`getRandomText(int numChars)`**
  - Type of parameter is meaningful, not name!
  - Add words not characters, manage spaces

```
public String getRandomText(int numWords){
    StringBuilder sb = new StringBuilder();
    int index = myRandom.nextInt(myText.length-1);
    String key = myText[index];
    sb.append(key);
    sb.append(" ");
    // for loop not shown
    return sb.toString().trim();
}
```

Completing MarkovWordOne

- Code is nearly identical to MarkovOne

```
for(int k=0; k < numWords-1; k++){
      ArrayList<String> follows = getFollows(key);
      if (follows.size() == 0){
        break;
      }
      index = myRandom.nextInt(follows.size());
      String next = follows.get(index);
      sb.append(next);
      sb.append(" ");
      key = next;
}
```

Completing MarkovWordOne

- Code is nearly identical to MarkovOne
  - Changed **`numChars`** to **`numWords`**

```
for(int k=0; k < numWords-1; k++){
      ArrayList<String> follows = getFollows(key);
      if (follows.size() == 0){
        break;
      }
      index = myRandom.nextInt(follows.size());
      String next = follows.get(index);
      sb.append(next);
      sb.append(" ");
      key = next;
}
```

Completing MarkovWordOne

- Code is nearly identical to MarkovOne
  - Changed **`numChars`** to **`numWords`**
  - Append " " to StringBuilder

```
for(int k=0; k < numWords-1; k++){
      ArrayList<String> follows = getFollows(key);
      if (follows.size() == 0){
        break;
      }
      index = myRandom.nextInt(follows.size());
      String next = follows.get(index);
      sb.append(next);
      sb.append(" ");
      key = next;
}
```

Coding and Testing

- We'll copy **`getFollows`** from MarkovOne
  - Searches String **`myText`**
  - String methods **`.length()`** and **`.indexOf()`**
  - Uses **`.substring()`** for one-character String
- Changes due to String[] **`myText`**
  - We'll need to write **`.indexOf()`** for arrays
  - Java doesn't supply indexing search, does supply **`.contains`** for ArrayList