Word N-Grams

WordGram Class

Duke
UNIVERSITY

Markov Models

- MarkovOne to MarkovTwo and more
  - for characters, changes are very simple: change 1 to 2 to **`myOrder`** in **`getRandomText`**

```
public String getRandomText(int length) {
    StringBuilder sb = new StringBuilder();
    int index = myRandom.nextInt(myText.length() - myOrder);
    String current = myText.substring(index, index + myOrder);
    sb.append(current);
    for(int k=0; k < length-myOrder; k++){
        ArrayList<String> follows = getFollows(current);
```

Markov Models

- MarkovOne to MarkovTwo and more
  - for characters, changes are very simple: change 1 to 2 to **`myOrder`** in **`getRandomText`**

```
public String getRandomText(int length) {
    StringBuilder sb = new StringBuilder();
    int index = myRandom.nextInt(myText.length() - myOrder);
    String current = myText.substring(index, index + myOrder);
    sb.append(current);
    for(int k=0; k < length-myOrder; k++){
        ArrayList<String> follows = getFollows(current);
```

Markov Models

- MarkovOne to MarkovTwo and more
  - for characters, changes are very simple: change 1 to 2 to **`myOrder`** in **`getRandomText`**

```
public String getRandomText(int length) {
    StringBuilder sb = new StringBuilder();
    int index = myRandom.nextInt(myText.length() - myOrder);
    String current = myText.substring(index, index + myOrder);
    sb.append(current);
    for(int k=0; k < length-myOrder; k++){
        ArrayList<String> follows = getFollows(current);
```

Markov Models

- MarkovOne to MarkovTwo and more
  - for characters, changes are very simple: change 1 to 2 to **`myOrder`** in **`getRandomText`**

```
public String getRandomText(int length) {
    StringBuilder sb = new StringBuilder();
    int index = myRandom.nextInt(myText.length() - myOrder);
    String current = myText.substring(index, index + myOrder);
    sb.append(current);
    for(int k=0; k < length-myOrder; k++){
        ArrayList<String> follows = getFollows(current);
```

Markov Models

- MarkovOne to MarkovTwo and more
  - for characters, changes are very simple: change 1 to 2 to **`myOrder`** in **`getRandomText`**
  - No changes to: **`getFollows()`**

```
public String getRandomText(int length) {
    StringBuilder sb = new StringBuilder();
    int index = myRandom.nextInt(myText.length() - myOrder);
    String current = myText.substring(index, index + myOrder);
    sb.append(current);
    for(int k=0; k < length-myOrder; k++){
        ArrayList<String> follows = getFollows(current);
```

Markov Models

- MarkovOne to MarkovTwo and more
  - for characters, changes are very simple: change 1 to 2 to **`myOrder`** in **`getRandomText`**
  - No changes to: **`getFollows()`**
- Changes to MarkovWordOne not so simple
  - Strings: character sequences —**`.substring`** gets any subsequence
  - Need analog for String array

Method getFollows for N chars

- String **`myText`**, String **`key`** has any length

```
private String myText;

protected ArrayList<String> getFollows(String key){

    // code not shown

    return follows;
}
```

Method getFollows for N chars

- String **`myText`**, String **`key`** has any length

```
private String myText;

protected ArrayList<String> getFollows(String key){

    // code not shown

    return follows;
}
```

Method getFollows for N chars

- String **`myText`**, String **`key`** has any length

```
private String myText;

protected ArrayList<String> getFollows(String key){

    // code not shown

    return follows;
}
```

Method getFollows for N chars

- String **`myText`**, String **`key`** has any length
  - **`.indexOf(..)`** and **`.substring(..)`** work

```
private String myText;

protected ArrayList<String> getFollows(String key){

    // code not shown

    return follows;
}
```

Extending to N Words?

- MarkovOne to MarkovTwo/N for characters
  - Simple because String key was 1, 2, 3, … characters
  - Key is a String, argument to **`.indexOf(..)`**
- MarkovWordOne to MarkovWord/N
  - One-word key to N-word key
  - How to search for N words in a String[]
- Design and implement word sequence
  - Analog of String as character sequence
  - We will call this WordGram

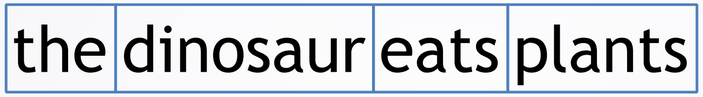
WordGram

- Sequence of Strings, not characters
  - String is "plant" or "dinosaur"

WordGram

- Sequence of Strings, not characters
  - String is "plant" or "dinosaur"

WordGram

- Sequence of Strings, not characters
  - String is "plant" or "dinosaur"
  - WordGram is 

- The type char is primitive; String is not
  - Internally String is array of char values
  - Internally WordGram is array of String values