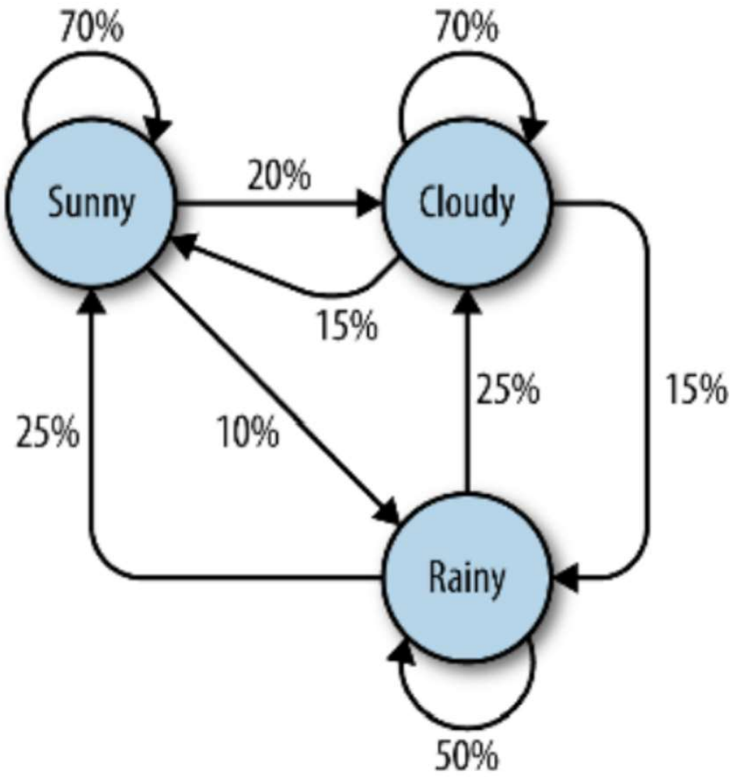
Markov Model

- If a day is Sunny,
  - Next day has 70% chances of being Sunny.
  - Next day has 20% chances of being cloudy
  - Next day has 10% chances of being rainy
- Markov model can be generated using historic data



Markov Model

- Probability of next word depends on previous word.
- Random numbers may be used to generate sentences.

- Example:
  - If current word is ‘Word 1’, with 50% chances, next word is ‘Word 2’, with 30% probability, the next word is ‘Word 3’, etc.

Parts of Speech used in NLTK

- These are given in the next page

CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential "there"
FW	Foreign word
IN	Preposition, subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	"to"
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-third-person singular present
VBZ	Verb, third person singular present
WDT	wh-determiner
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb

Chapter 8: Cleaning your dirty data

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
html=urlopen('https://en.wikipedia.org/wiki/Python_(programming_language)')
bs=BeautifulSoup(html,'html.parser')
print(bs.find().get_text())
```

```
Python (programming language) - Wikipedia
document.documentElement.className="client-js";RLCONF={"wgBreakFrames":false,"wgSeparatorTransformTable":["",""],"wgDigitTran
"All articles containing potentially dated statements","Articles containing potentially dated statements from October 2021","
"Programming languages","Programming languages created in 1991","Scripting languages","Text-oriented programming languages"],
"wgWikibaseItemId":"Q28865","wgGENewcomerTasksGuidanceEnabled":true,"wgGEAskQuestionEnabled":false,"wgGELinkRecommendationsFr
"ext.wikimediaEvents","ext.navigationTiming","ext.cx.eventlogging.campaigns","ext.quicksurveys.init","ext.centralNotice.geoIP
(RLQ=window.RLQ||[]).push(function(){mw.loader.implement("user.options@1i9g4",function($,jQuery,require,module){mw.user.token
```

```
Python (programming language)

From Wikipedia, the free encyclopedia
```

Cleaning using regular expression

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import re
html=urlopen('https://en.wikipedia.org/wiki/Python_(programming_language)')
bs=BeautifulSoup(html,'html.parser')
t=bs.find().get_text()
```

```
t=re.sub('\n|[[\d+\]]',' ',t)
print(t)
```

```
   Python (programming language) - Wikipedia document.documentElement.className="client-js";RLCONF={"wgBreakFrames":false,"wgSe
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: FutureWarning: Possible nested set at position 3
  import sys
```

Processing the text

n-gram

In linguistics and Natural Language Processing, an n-gram is a sequence of n words in text or speech.

Sentence:'I like this book'

2-gram: [['I', 'like'], ['like', 'this'], ['this', 'book']]

3-gram: [['I','like','this'],['like','this','book']]

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
def genNgrams(content, n):
  content=content.split(' ')
  output=[]
  for i in range(len(content)-n+1):
    output.append(content[i:i+n])
  return output
html=urlopen('https://en.wikipedia.org/wiki/Python_(programming_language)')
bs=BeautifulSoup(html,'html.parser')
#content=bs.find().get_text()
content=bs.find('div',{'id':'mw-content-text'}).get_text()
#content=re.sub('\n|[[\d+\]]',' ',t)
ngrams=genNgrams(content,2)
print(ngrams)
print('Number of 2-grams is ',len(ngrams))
```

```
[['General-purpose', 'programming'], ['programming', 'language\n\n\n.mw-parser-output'], ['language\n\n\n.mw-parser-output', '.
Number of 2-grams is  11702
```

Cleaning it and then obtaining n-grams

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
def genNgrams(content, n):

  content=re.sub('\n|[[\d+\]]',' ',content)
  content=bytes(content,'UTF-8')
  content=content.decode('ascii','ignore')
  content=content.split(' ')
  content=[word for word in content if word!='']
  output=[]
  for i in range(len(content)-n+1):
    output.append(content[i:i+n])
  return output
html=urlopen('https://en.wikipedia.org/wiki/Python_(programming_language)')
bs=BeautifulSoup(html,'html.parser')
#content=bs.find().get_text()
content=bs.find('div',{'id':'mw-content-text'}).get_text()

ngrams=genNgrams(content,2)
print(ngrams)
print('Number of 2-grams is ',len(ngrams))
```

```
[['General-purpose', 'programming'], ['programming', 'language'], ['language', '.mw-parser-output'], ['.mw-parser-output', '.in
Number of 2-grams is  12702
```

Install nltk

import nltk

nltk.download()

When asked for packages, select 'all'

```
import nltk
nltk.download()
```

```
    NLTK Downloader
    ---------------------------------------------------------------------------
        d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
    ---------------------------------------------------------------------------
    Downloader> d

    Download which package (l=list; x=cancel)?
      Identifier> l
    Packages:
      [ ] abc................. Australian Broadcasting Commission 2006
      [ ] alpino.............. Alpino Dutch Treebank
      [ ] averaged_perceptron_tagger Averaged Perceptron Tagger
      [ ] averaged_perceptron_tagger_ru Averaged Perceptron Tagger (Russian)
      [ ] basque_grammars..... Grammars for Basque
      [ ] biocreative_ppi..... BioCreAtIvE (Critical Assessment of Information
                               Extraction Systems in Biology)
      [ ] bllip_wsj_no_aux.... BLLIP Parser: WSJ Model
      [ ] book_grammars....... Grammars from NLTK Book
      [ ] brown............... Brown Corpus
      [ ] brown_tei........... Brown Corpus (TEI XML Version)
      [ ] cess_cat............ CESS-CAT Treebank
      [ ] cess_esp............ CESS-ESP Treebank
      [ ] chat80.............. Chat-80 Data Files
      [ ] city_database....... City Database
      [ ] cmudict............. The Carnegie Mellon Pronouncing Dictionary (0.6)
      [ ] comparative_sentences Comparative Sentence Dataset
      [ ] comtrans............ ComTrans Corpus Sample
      [ ] conll2000........... CONLL 2000 Chunking Corpus
      [ ] conll2002........... CONLL 2002 Named Entity Recognition Corpus
    Hit Enter to continue:
      [ ] conll2007........... Dependency Treebanks from CoNLL 2007 (Catalan
```

```
                        and Basque Subset)
  [ ] crubadan.......... Crubadan Corpus
  [ ] dependency_treebank. Dependency Parsed Treebank
  [ ] dolch............. Dolch Word List
  [ ] europarl_raw...... Sample European Parliament Proceedings Parallel
                        Corpus
  [ ] extended_omw...... Extended Open Multilingual WordNet
  [ ] floresta.......... Portuguese Treebank
  [ ] framenet_v15...... FrameNet 1.5
  [ ] framenet_v17...... FrameNet 1.7
  [ ] gazetteers........ Gazeteer Lists
  [ ] genesis........... Genesis Corpus
  [ ] gutenberg......... Project Gutenberg Selections
  [ ] ieer.............. NIST IE-ER DATA SAMPLE
  [ ] inaugural......... C-Span Inaugural Address Corpus
  [ ] indian............ Indian Language POS-Tagged Corpus
  [ ] jeita............. JEITA Public Morphologically Tagged Corpus (in
                        ChaSen format)
  [ ] kimmo............. PC-KIMMO Data Files
  [ ] knbc.............. KNB Corpus (Annotated blog corpus)
Hit Enter to continue:
  [ ] large_grammars..... Large context-free and feature-based grammars
                        for parser comparison
  [ ] lin_thesaurus..... Lin's Dependency Thesaurus
  [ ] mac_morpho........ MAC-MORPHO: Brazilian Portuguese news text with
                        part-of-speech tags
  [ ] machado           Machado de Assis -- Obra Completa
```

Data cleaning

```
import string
sentence='This is a  book. It is a      weg.'
sentence=sentence.split(' ')
sentence=[word.strip(string.punctuation+string.whitespace) for word in sentence]
sentence=[word for word in sentence if len(word)>1 or (word.lower()=='a' or word.lower=='i')]
print(sentence)
```

```
['This', 'is', 'a', 'book', 'It', 'is', 'a', 'weg']
```

More cleaning

```
import re
sentence='This is a book? It is\n\n\ndfsjfjdkjk q[123]q \a\b'
content=re.sub('\n|[[\d+\]]',' ',sentence)
print(content)
content=bytes(content,'UTF-8')
print(content)
content=content.decode('ascii','ignore')
print(content)
```

```
This is a book? It is   dfsjfjdkjk q     q
b'This is a book? It is   dfsjfjdkjk q     q \x07\x08'
This is a book? It is   dfsjfjdkjk q     q
```

```
print(string.punctuation)
```

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

Extracting sentences

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import re
import string
def cleanSentence(sentence):
  sentence=sentence.split(' ')
  sentence=[word.strip(string.punctuation+string.whitespace) for word in sentence]
  sentence=[word for word in sentence if len(word)>1 or (word.lower()=='a' or word.lower=='i')]
  return sentence
def cleanInput(content):
  content=re.sub('\n|[[\d+\]]',' ', content)
  content=bytes(content,'UTF-8')
  content=content.decode('ascii','ignore')
  sentences=content.split('. ')
  return [cleanSentence(sentence) for sentence in sentences]
def getNgramsFromSentences(content,n):
  output=[]
  for i in range(len(content)-n+1):
    output.append(content[i:i+n])
  return output
def getNgrams(content,n):
  content=cleanInput(content)
  ngrams=[]
  for sentence in content:
    ngrams.extend(getNgramsFromSentences(sentence,n))
  return ngrams
```

Generating ngrams from the above code

```
html=urlopen('https://en.wikipedia.org/wiki/Python_(programming_language)')
bs=BeautifulSoup(html,'html.parser')
```

```
content=bs.find('div',{'id':'mw-content-text'}).get_text()
ngrams=getNgrams(content,2)
print(ngrams)
print('2-gram count is ',len(ngrams))
```

```
[['General-purpose', 'programming'], ['programming', 'language'], ['language', 'mw-parser-output'], ['mw-parser-output', 'infob
2-gram count is  9591
```

Finding the number of occurances of n-grams

```
from collections import Counter
def getNgrams(content,n):
  content=cleanInput(content)
  ngrams=Counter()
  for sentence in content:
    newNgrams=[' '.join(ngram) for ngram in getNgramsFromSentences(sentence,2)]
    ngrams.update(newNgrams)
  return ngrams
print(getNgrams(content,2))
```

```
Counter({'from the': 218, 'the original': 209, 'original on': 207, 'Archived from': 200, 'on June': 60, 'Software Foundation':
```