N-Gram Modelling

Markov chains - States:

- A is a state
- B is a state

- Probability of A → B : 50%.
- Probability of A → A : 50%.
- Probabity of B → A : 50%.
- Probability of B → B : 50%.

AABA

Seq of chain of states is called Markov chains.

N-Gram

"An N-gram is a Contiguous sequence of n items from a given sample of text & speech" → wikipedia

Items refer to states in Markov chains

Items can be "characters", 'words', 'sentences" Etc.,

N = 2, bigram.

N = 3 Trigrams

characters are the states of characters chains

N = 2 "the bird is flying on the blue sky"

Bigrams = 'th', 'he', 'e', 'b', 'bi', 'ir', 'rd', 'd', 'i', etc.,

N = 3

Trigrams = 'the', 'he*', 'e b', 'bi' ...etc.,

Trigrams	next
the	→
he	→ b
e b	→ i
bi	→ r
bir	→ d
ird	→

Word Grams

"The bird is flying on the sky"

Trigram	Next
The bird is	flying
bird is flying	on
is flying on	the
flying on the	blue
on the blue	sky
The blue sky	→

Next (Item when large corpus)

[flying, eating, sleeping]

[on, through, on]

The

[blue, orange]

sky

[.]

→ If we are looking at a huge corpus

Application

→ Auto complete system

The bird is [flying, eating, sleeping]

* Select any word at random.

→ The bird is flying [on, through, on]

→ The bird is flying on [the]

→ The bird is flyin on the [blue, orange]

→ The bird is flyin on the blue sky.

→ Ners are used in the smart phones, Google

→ Entertainment, Entertain.

http://text-analytics101.rxnlp.com/2014/11/what-are-n-grams.html

Code N gram (Character n gram model)

```
text = """ Global warming """

n = 3

ngrams = {}

# Create n grams

for i in range(len(text) - n):

    gram = text[i : i+n]   # text[0:3] = Glo

    If gram not in ngrams.keys()

        ngrams[gram] = [ ]

    ngrams[gram].appen(text[i: i+n])

# testing our N-gram model

CurrentGram = text[0:n]

result = CurrentGram

for i in range(100):

    If CurrentGram not in ngrams.keys():

        break

    Possibilities = ngrams[CurrentGram]

    nextitem = Possibilities[random.randrange(len(Possibilities))]

    result += nextitem

    Currentgram = result[len(result) - n: len(result)]

Print (result)
```

Code for N Gram word

```
text = """ Global warming & climate change has become a world wide concern"""
import random
n = 3
ngrams = { }
words = nltk.words.tokenize(text)
for i in range(len(words)-n):
    gram = ' '.join(words[i:i+n])
    If gram not in ngrams.keys():
        ngram[gram] = [ ]
    ngram[gram].append(words[i+n])
&
# testing our N-gram model
    CurrentGram = ' '.join(words[0:n])
    result = Currentgram.
    for i in range(30):
        if currentGram not in ngrams.keys():
            break
        possibilities = ngrams[currentgram]
        nextitem = possibilities[random.randrange(len(possibilities))]
        result += ' ' + nextitem
        currentgram = result[len(result)-n):len(result)]
        rwords = nltk.word_tokenize(result)
        currentgram = ' '.join(rwords[len(rwords)-n:len(rwords)])
print(result)
```