Problem Statement : When studying a system that can change over time, we need a way to track those changes. And those changes may or may not be guided by probabilistic methods. However, in order to predict the future state of that system, we can use probabilities in count to bias our prediction, allowing us to predict the next state with the most possibilities. Often, directly inferring values is not tractable with probabilistic models, necessitating the use of approximation methods.

System Requirements :

● Processor	: Intel(R) Core(TM) i3-7100U CPU @ 2.40GHz 2.40 GHz
● RAM	: 4.00 GB
● Operating System	: Windows 10
● IDE	: Jupyter Notebook

Algorithm : Markov Chain

Markov chains, named after Andrey Markov, are a stochastic model that depicts a sequence of possible events with predictions or probabilities for the next state based solely on the previous event state, rather than the states before. In other words, the probability that the n+1 th steps will be x is determined solely by the nth steps and not by the entire sequence of steps preceding n. This is known as the Markov Property or Memorylessness.
Formal definition of discrete chain Markov chain can represented as:

$$P(Xt+1 = s \mid Xt = st , Xt-1 = st-1, . . . , X0 = s0) = P(Xt+1 = s \mid Xt = st)$$

Let {X0, X1, X2, . . .} be a sequence of discrete random variables. Then {X0, X1, X2, . . .} is a Markov chain if it satisfies the Markov property: P(Xt+1 = s | Xt = st , . . . , X0 = s0) = P(Xt+1 = s | Xt = st), for all t = 1, 2, 3, . . . and for all states s0, s1, . . . , st , s.
(Xt+1) depends only on Xt . It does not depend upon X0, X1, . . . , Xt−1.
In our project, we took the script of the "*Harry Potter and the Philosopher's Stone*" movie as our dataset. And the occurrence of the names of the most commonly used characters are the states of this problem. We took four names as states, and those are '*harry*', '*hermione*', '*ron*' and '*hagrid*'. So our Markov Chain is predicting the future occurrences of any character's name in between these four names. In this script, '*harry*' was called 1136 times,'*ron*': 417 times, '*hagrid*': 332 times and '*hermione' :* 281 times. So we took only these four characters names and made our occurrence list to have the transition matrix.
The matrix describing the Markov chain is called the transition matrix. It is the most important tool for analysing Markov chains. The transition matrix is usually given the symbol P = (pij).

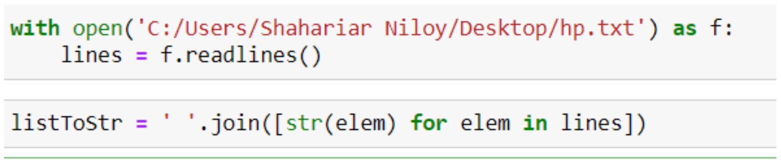
$$pij = P(Xt+1 = j \mid Xt = i)$$

The transition matrix P must list all possible states in the state space. P is a square matrix (N × N), because Xt+1 and Xt both take values in the same state space. The rows of P should each sum to 1.

After generating the transition matrix, we build the markov chain and generate 5 random walks having a random number of states in between 50 to 100.

Implementations :

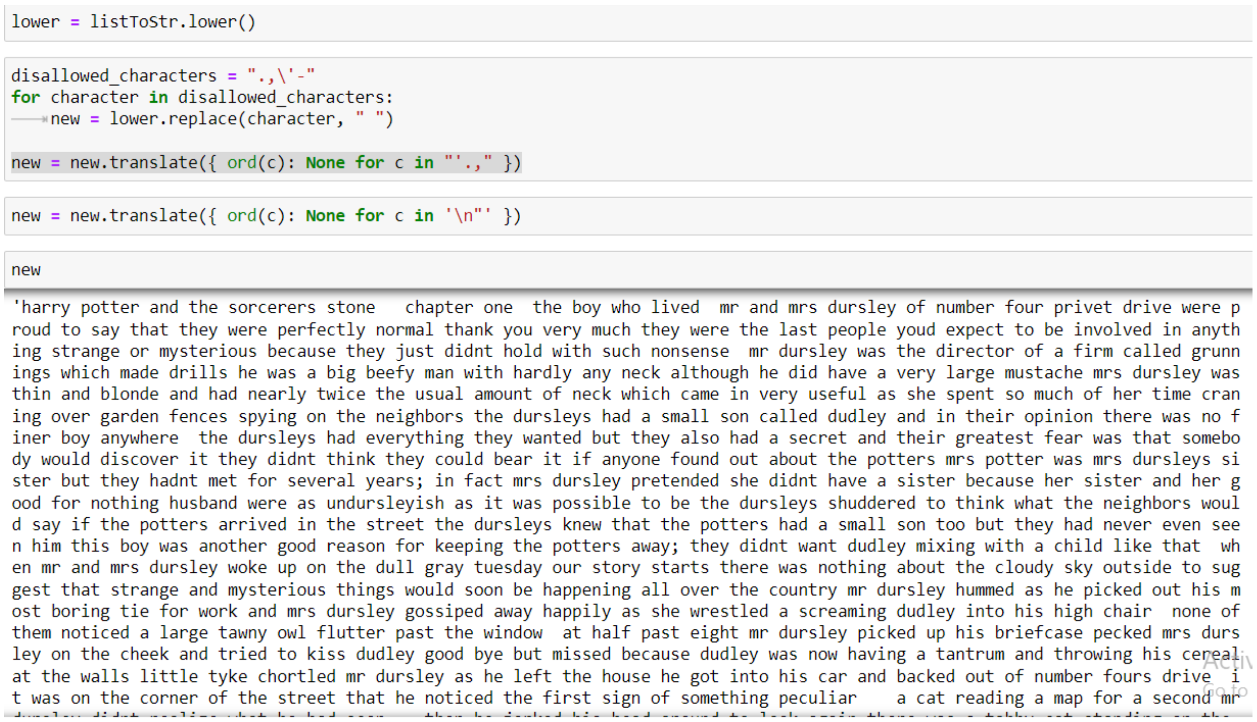
- **Preprocessing :** After reading the dataset, we convert our dataset list into string. And from that we change the whole script into lowercase words and eliminate special characters so that we can get our states name in a good form.



```
with open('C:/Users/Shahariar Niloy/Desktop/hp.txt') as f:
    lines = f.readlines()
```

```
listToStr = ' '.join([str(elem) for elem in lines])
```

Figure: 01



```
lower = listToStr.lower()
```

```
disallowed_characters = ".,\'-"
for character in disallowed_characters:
    new = lower.replace(character, " ")

new = new.translate({ ord(c): None for c in "'.," })
```

```
new = new.translate({ ord(c): None for c in '\n"' })
```

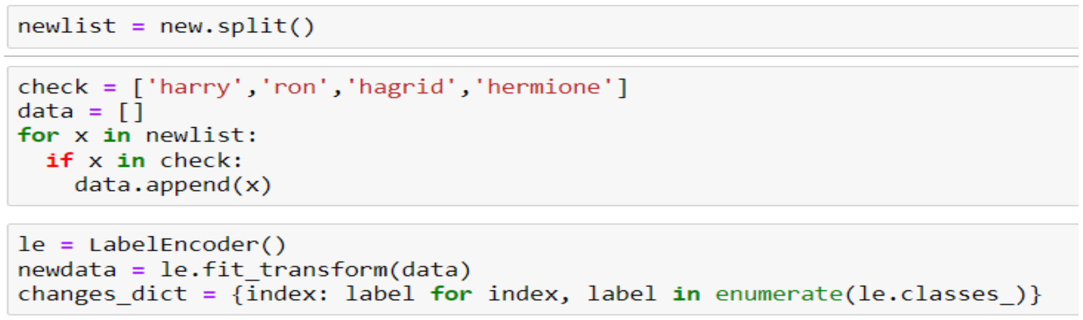
```
new
```

```
'harry potter and the sorcerers stone   chapter one  the boy who lived  mr and mrs dursley of number four privet drive were p
roud to say that they were perfectly normal thank you very much they were the last people youd expect to be involved in anyth
ing strange or mysterious because they just didnt hold with such nonsense  mr dursley was the director of a firm called grunn
ings which made drills he was a big beefy man with hardly any neck although he did have a very large mustache mrs dursley was
thin and blonde and had nearly twice the usual amount of neck which came in very useful as she spent so much of her time cran
ing over garden fences spying on the neighbors the dursleys had a small son called dudley and in their opinion there was no f
iner boy anywhere  the dursleys had everything they wanted but they also had a secret and their greatest fear was that somebo
dy would discover it they didnt think they could bear it if anyone found out about the potters mrs potter was mrs dursleys si
ster but they hadnt met for several years; in fact mrs dursley pretended she didnt have a sister because her sister and her g
ood for nothing husband were as undursleyish as it was possible to be the dursleys shuddered to think what the neighbors woul
d say if the potters arrived in the street the dursleys knew that the potters had a small son too but they had never even see
n him this boy was another good reason for keeping the potters away; they didnt want dudley mixing with a child like that  wh
en mr and mrs dursley woke up on the dull gray tuesday our story starts there was nothing about the cloudy sky outside to sug
gest that strange and mysterious things would soon be happening all over the country mr dursley hummed as he picked out his m
ost boring tie for work and mrs dursley gossiped away happily as she wrestled a screaming dudley into his high chair  none of
them noticed a large tawny owl flutter past the window  at half past eight mr dursley picked up his briefcase pecked mrs durs
ley on the cheek and tried to kiss dudley good bye but missed because dudley was now having a tantrum and throwing his cereal
at the walls little tyke chortled mr dursley as he left the house he got into his car and backed out of number fours drive  i
t was on the corner of the street that he noticed the first sign of something peculiar    a cat reading a map for a second mr
```

Figure: 02

Here, after reading the ‘hp.txt’ file, the script was stored into a list. We convert the list into string so that we can remove special characters like [, . ‘ \n]. ‘new ’ string is the complete preprocessed data.

- **Creating States :** From the ‘new’ string of the script, we convert each word into a list item and check the occurrence of those names which are in the states list. And create a list of the names of those four characters having the same sequence of occurrences in the script.
  Then we encoded those names into a numeric form for getting help to make a transition matrix using labelencoder. Alongside, we keep the track of changes into a dictionary, so that we can understand digit representation.

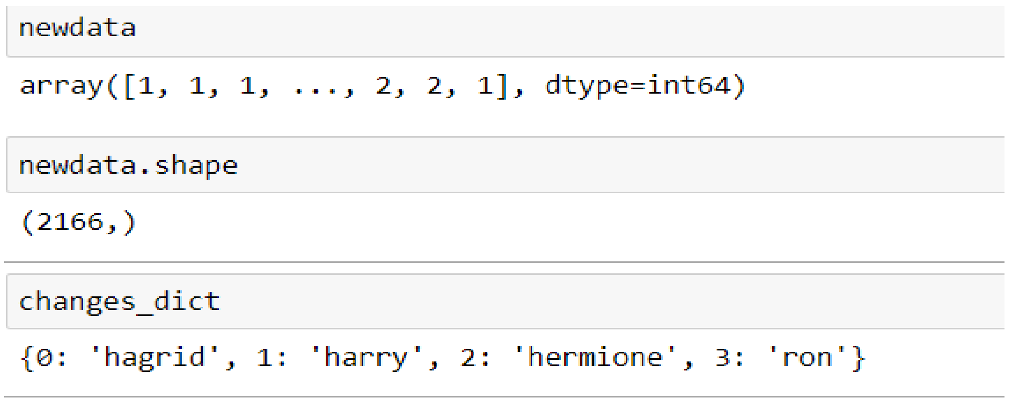


```
newlist = new.split()
```

```
check = ['harry','ron','hagrid','hermione']
data = []
for x in newlist:
  if x in check:
    data.append(x)
```

```
le = LabelEncoder()
newdata = le.fit_transform(data)
changes_dict = {index: label for index, label in enumerate(le.classes_)}
```

Figure: 03



```
newdata
```

```
array([1, 1, 1, ..., 2, 2, 1], dtype=int64)
```

```
newdata.shape
```

```
(2166,)
```

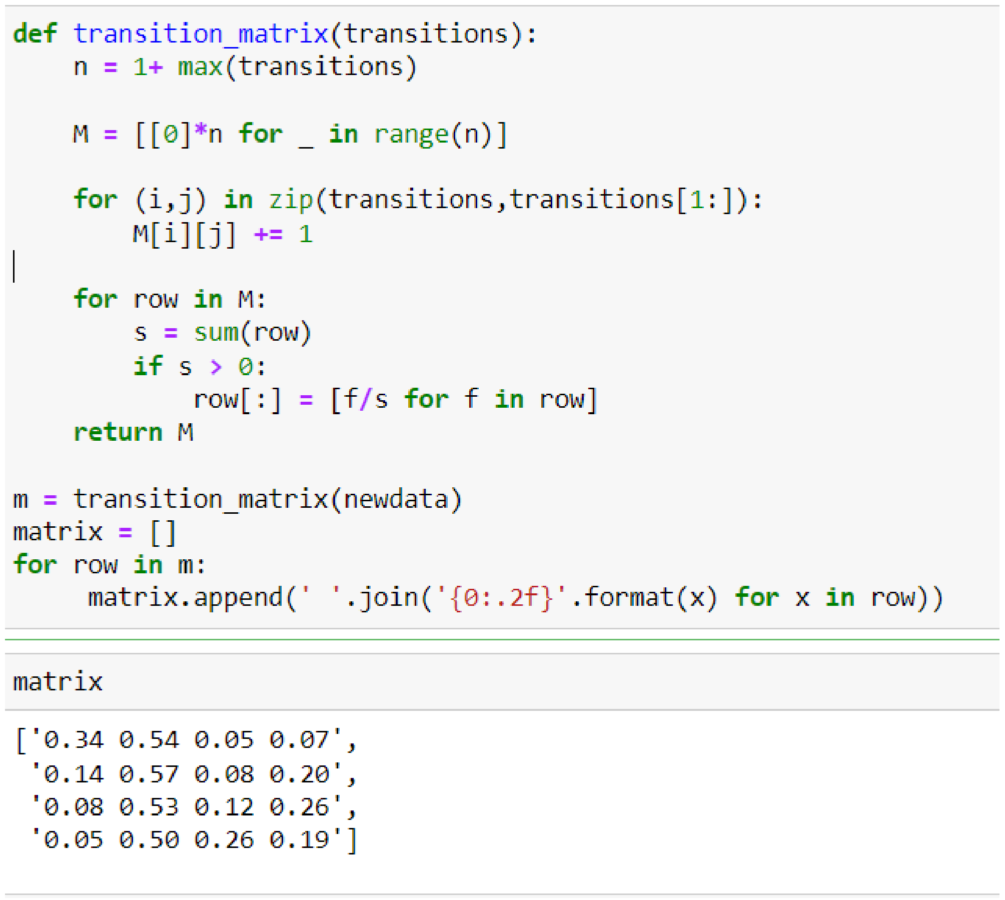
```
changes_dict
```

```
{0: 'hagrid', 1: 'harry', 2: 'hermione', 3: 'ron'}
```

Figure: 04

Here, our state vector is [‘*hagrid*’, ‘*harry*’, ‘*hermione*’, ‘*ron*’]. And we change these names into [0 1 2 3] format. Our states array has 2166 records and changes_dict navigating the changes after using labelencoder.

- **Transition Matrix :** Transition matrix is one of the key components of Markov-Chain. To build this matrix we pass the data array into a function called ‘transition_matrix’, where the function gets the input and counts the immediate occurrences of those numbers and sum up the count values and store them into a (n X n) matrix. Before returning the matrix, that matrix was converted into probabilities.



```python
def transition_matrix(transitions):
    n = 1+ max(transitions)

    M = [[0]*n for _ in range(n)]

    for (i,j) in zip(transitions,transitions[1:]):
        M[i][j] += 1

    for row in M:
        s = sum(row)
        if s > 0:
            row[:] = [f/s for f in row]
    return M

m = transition_matrix(newdata)
matrix = []
for row in m:
     matrix.append(' '.join('{0:.2f}'.format(x) for x in row))
```

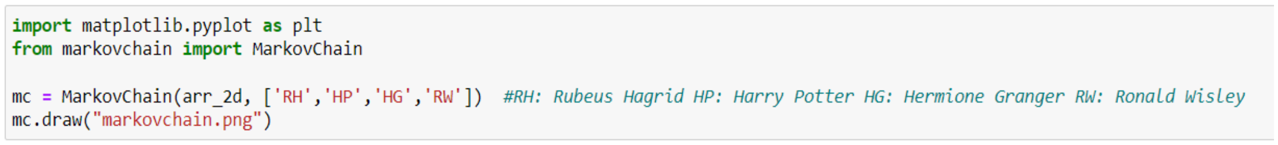
```python
matrix
```

```
['0.34 0.54 0.05 0.07',
 '0.14 0.57 0.08 0.20',
 '0.08 0.53 0.12 0.26',
 '0.05 0.50 0.26 0.19']
```

Figure: 05

Here, ‘newdata’ is an argument to the transition_matrix() function and ‘matrix’ is the transition matrix of this markov chain.

- **Markov-Chain:** We plot a markov-chain diagram with corresponding probabilities, so that we can visualize the algorithm and best possible next state.



```python
import matplotlib.pyplot as plt
from markovchain import MarkovChain

mc = MarkovChain(arr_2d, ['RH','HP','HG','RW'])  #RH: Rubeus Hagrid HP: Harry Potter HG: Hermione Granger RW: Ronald Wisley
mc.draw("markovchain.png")
```

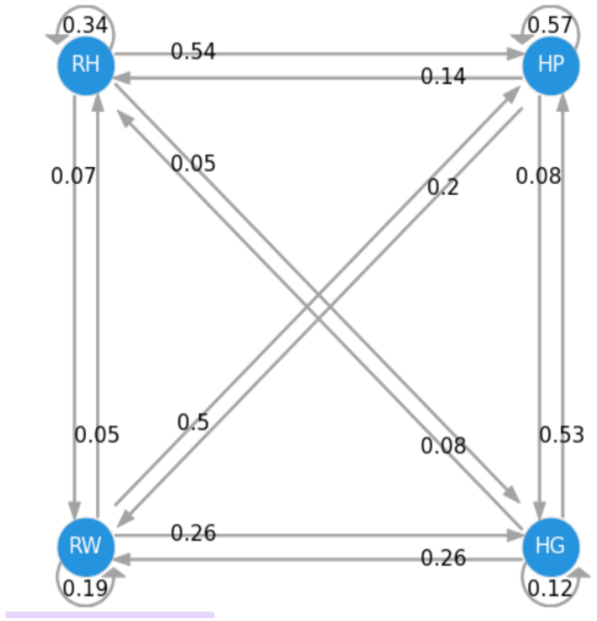


Figure: 06

Here, we plot the diagram using matplotlib along with the help of two other modules named '*node.py*' & '*markovchain.py*' . We got this module from the github repository which is given free to use by the author.

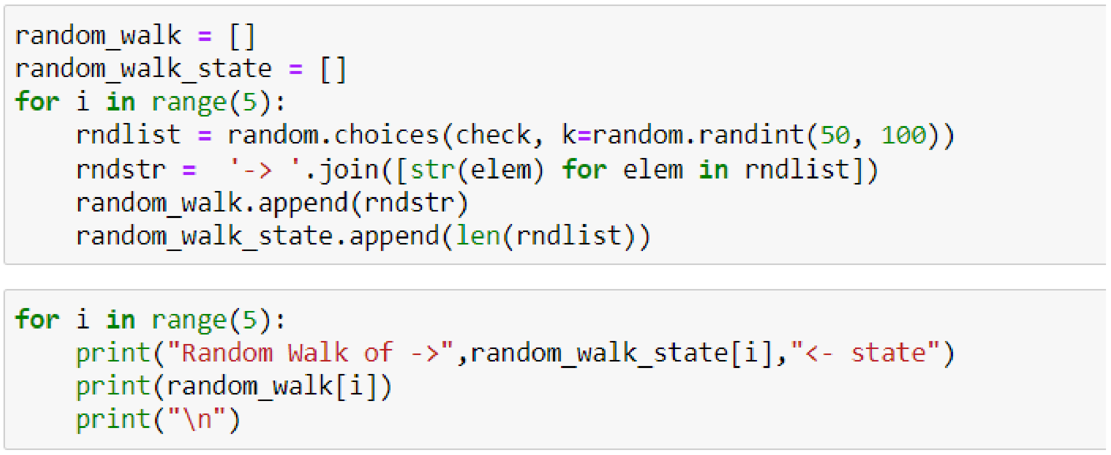
Github Repository : https://github.com/NaysanSaran/markov-chain

We use '*node.py*' & '*markovchain.py*', these two modules by importing MarkovChain, which is an author written module. We change the states name into :

- RH: Rubeus Hagrid (hagrid)
- HP: Harry Potter (harry)
- HG: Hermione Granger (hermione)
- RW: Ronald Wiesley (ron)

This diagram shows states and corresponding probabilities from one state to another.

- **Random Walk :** 5 different random walks are generated from states list using random function. And each random-walk contains a random number of steps (more than 50).

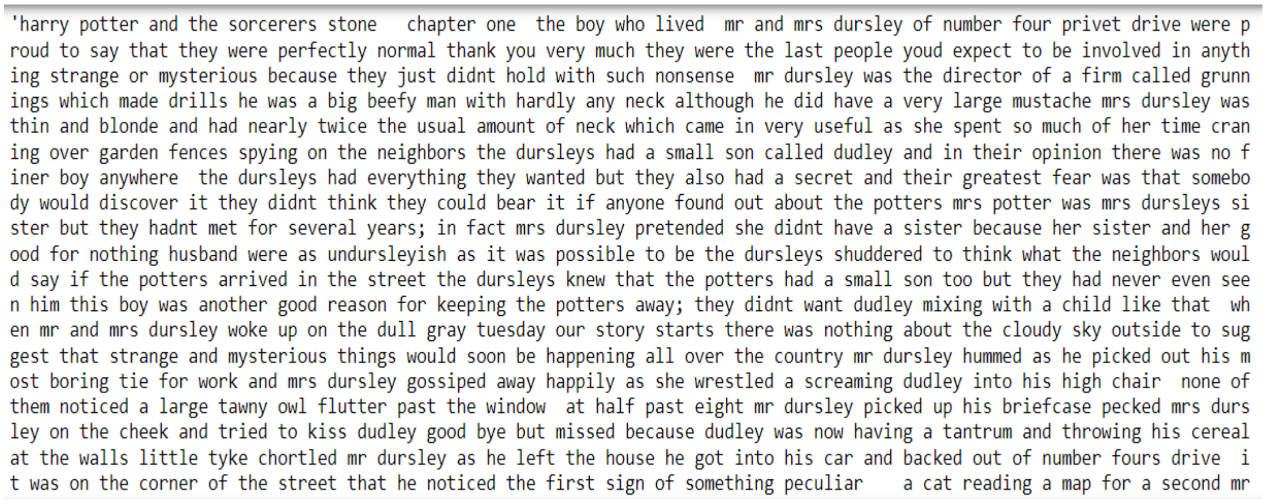


```python
random_walk = []
random_walk_state = []
for i in range(5):
    rndlist = random.choices(check, k=random.randint(50, 100))
    rndstr =  '-> '.join([str(elem) for elem in rndlist])
    random_walk.append(rndstr)
    random_walk_state.append(len(rndlist))
```

```python
for i in range(5):
    print("Random Walk of ->",random_walk_state[i],"<- state")
    print(random_walk[i])
    print("\n")
```

Figure: 07

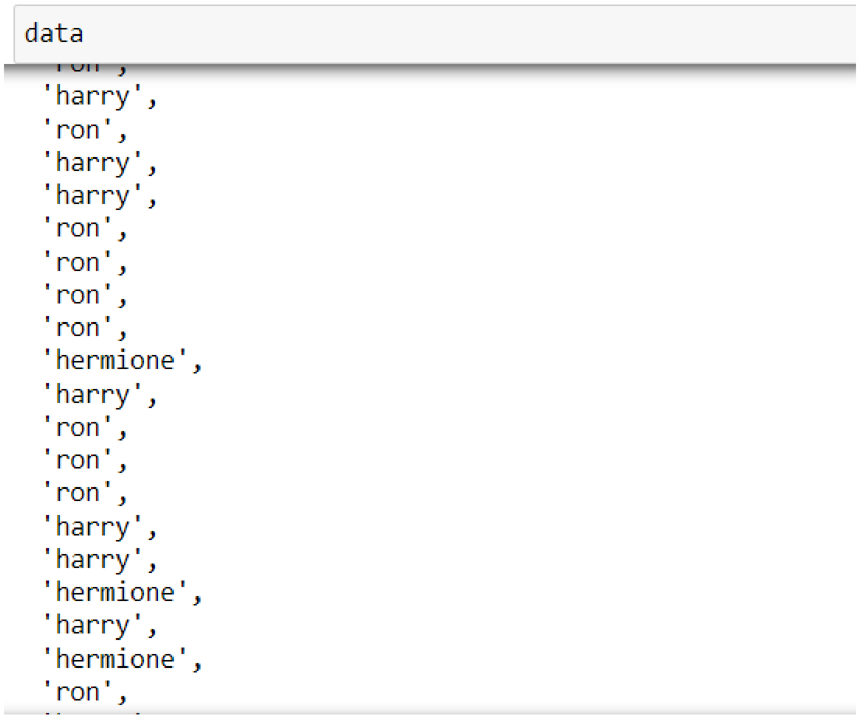
Testing Result: Our dataset was a movie script and after several preprocessing we converted the script into a good to use form.



```
'harry potter and the sorcerers stone   chapter one  the boy who lived  mr and mrs dursley of number four privet drive were p
roud to say that they were perfectly normal thank you very much they were the last people youd expect to be involved in anyth
ing strange or mysterious because they just didnt hold with such nonsense  mr dursley was the director of a firm called grunn
ings which made drills he was a big beefy man with hardly any neck although he did have a very large mustache mrs dursley was
thin and blonde and had nearly twice the usual amount of neck which came in very useful as she spent so much of her time cran
ing over garden fences spying on the neighbors the dursleys had a small son called dudley and in their opinion there was no f
iner boy anywhere  the dursleys had everything they wanted but they also had a secret and their greatest fear was that somebo
dy would discover it they didnt think they could bear it if anyone found out about the potters mrs potter was mrs dursleys si
ster but they hadnt met for several years; in fact mrs dursley pretended she didnt have a sister because her sister and her g
ood for nothing husband were as undursleyish as it was possible to be the dursleys shuddered to think what the neighbors woul
d say if the potters arrived in the street the dursleys knew that the potters had a small son too but they had never even see
n him this boy was another good reason for keeping the potters away; they didnt want dudley mixing with a child like that  wh
en mr and mrs dursley woke up on the dull gray tuesday our story starts there was nothing about the cloudy sky outside to sug
gest that strange and mysterious things would soon be happening all over the country mr dursley hummed as he picked out his m
ost boring tie for work and mrs dursley gossiped away happily as she wrestled a screaming dudley into his high chair  none of
them noticed a large tawny owl flutter past the window  at half past eight mr dursley picked up his briefcase pecked mrs durs
ley on the cheek and tried to kiss dudley good bye but missed because dudley was now having a tantrum and throwing his cereal
at the walls little tyke chortled mr dursley as he left the house he got into his car and backed out of number fours drive  i
t was on the corner of the street that he noticed the first sign of something peculiar    a cat reading a map for a second mr
```

Figure: 08

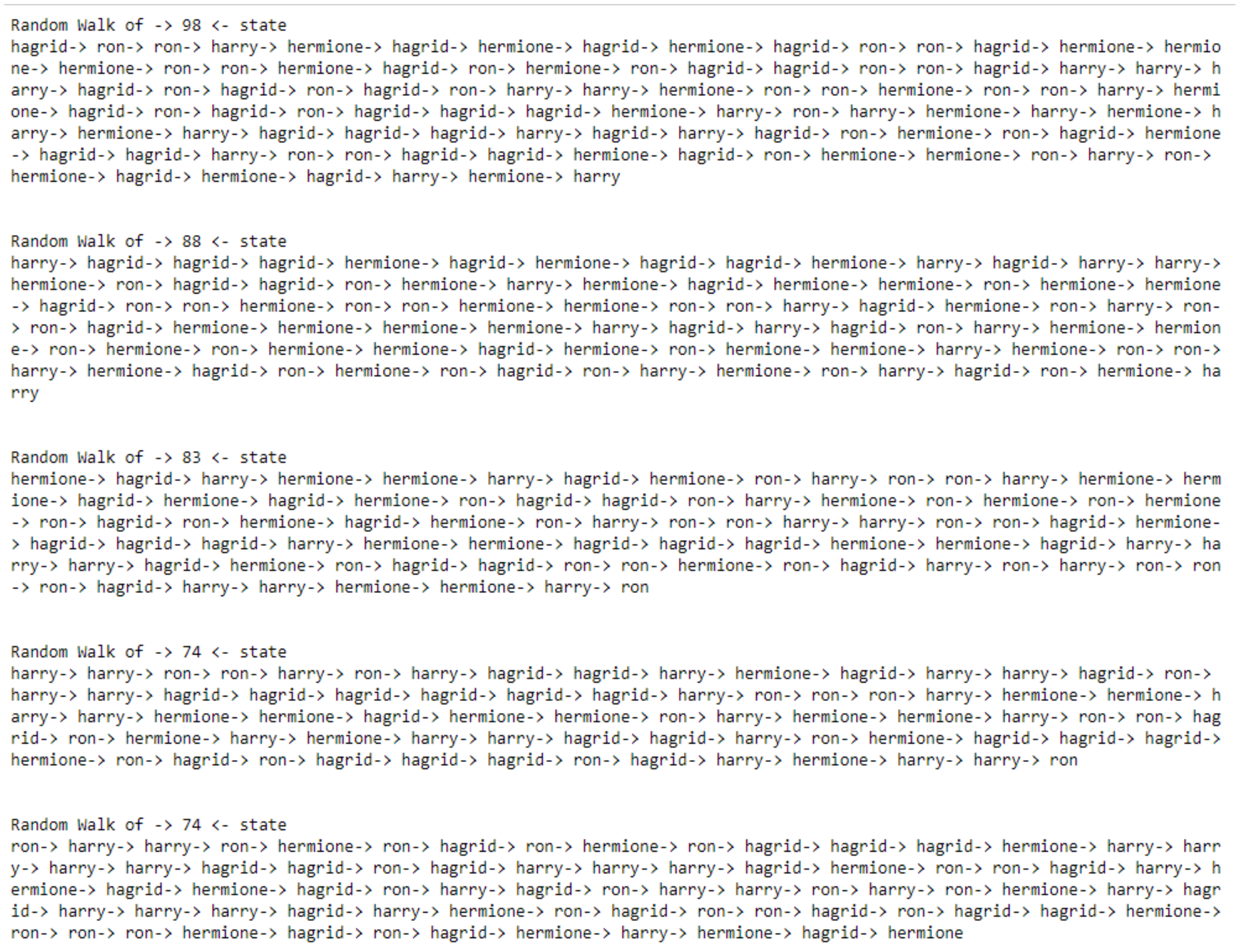
But from here we created another vector of states named '*data*', where selected states took place one after another following the script sequence.



```
data
 'harry',
 'ron',
 'harry',
 'harry',
 'ron',
 'ron',
 'ron',
 'ron',
 'hermione',
 'harry',
 'ron',
 'ron',
 'ron',
 'harry',
 'harry',
 'hermione',
 'harry',
 'hermione',
 'ron',
```

Figure: 09

After that we generated 5 random walks from this state vector, where it shows the possible next character name appearing in the current one.



```
Random Walk of -> 98 <- state
hagrid-> ron-> ron-> harry-> hermione-> hagrid-> hermione-> hagrid-> hermione-> hagrid-> ron-> ron-> hagrid-> hermione-> hermio
ne-> hermione-> ron-> ron-> hermione-> hagrid-> ron-> hermione-> ron-> hagrid-> hagrid-> ron-> ron-> hagrid-> harry-> harry-> h
arry-> hagrid-> ron-> hagrid-> ron-> hagrid-> ron-> harry-> harry-> hermione-> ron-> ron-> hermione-> ron-> ron-> harry-> hermi
one-> hagrid-> ron-> hagrid-> ron-> hagrid-> hagrid-> hagrid-> hermione-> harry-> ron-> harry-> hermione-> harry-> hermione-> h
arry-> hermione-> harry-> hagrid-> hagrid-> hagrid-> harry-> hagrid-> harry-> hagrid-> ron-> hermione-> ron-> hagrid-> hermione
-> hagrid-> hagrid-> harry-> ron-> ron-> hagrid-> hagrid-> hermione-> hagrid-> ron-> hermione-> hermione-> ron-> harry-> ron->
hermione-> hagrid-> hermione-> hagrid-> harry-> hermione-> harry


Random Walk of -> 88 <- state
harry-> hagrid-> hagrid-> hagrid-> hermione-> hagrid-> hermione-> hagrid-> hagrid-> hermione-> harry-> hagrid-> harry-> harry->
hermione-> ron-> hagrid-> hagrid-> ron-> hermione-> harry-> hermione-> hagrid-> hermione-> hermione-> ron-> hermione-> hermione
-> hagrid-> ron-> ron-> hermione-> ron-> ron-> hermione-> hermione-> ron-> ron-> harry-> hagrid-> hermione-> ron-> harry-> ron-
> ron-> hagrid-> hermione-> hermione-> hermione-> hermione-> harry-> hagrid-> harry-> hagrid-> ron-> harry-> hermione-> hermion
e-> ron-> hermione-> ron-> hermione-> hermione-> hagrid-> hermione-> ron-> hermione-> hermione-> harry-> hermione-> ron-> ron->
harry-> hermione-> hagrid-> ron-> hermione-> ron-> hagrid-> ron-> harry-> hermione-> ron-> harry-> hagrid-> ron-> hermione-> ha
rry


Random Walk of -> 83 <- state
hermione-> hagrid-> harry-> hermione-> hermione-> harry-> hagrid-> hermione-> ron-> harry-> ron-> ron-> harry-> hermione-> herm
ione-> hagrid-> hermione-> hagrid-> hermione-> ron-> hagrid-> hagrid-> ron-> harry-> hermione-> ron-> hermione-> ron-> hermione
-> ron-> hagrid-> ron-> hermione-> hagrid-> hermione-> ron-> harry-> ron-> ron-> harry-> harry-> ron-> ron-> hagrid-> hermione-
> hagrid-> hagrid-> hagrid-> harry-> hermione-> hermione-> hagrid-> hagrid-> hagrid-> hermione-> hermione-> hagrid-> harry-> ha
rry-> harry-> hagrid-> hermione-> ron-> hagrid-> hagrid-> ron-> ron-> hermione-> ron-> hagrid-> harry-> ron-> harry-> ron-> ron
-> ron-> hagrid-> harry-> harry-> hermione-> hermione-> harry-> ron


Random Walk of -> 74 <- state
harry-> harry-> ron-> ron-> harry-> ron-> harry-> hagrid-> hagrid-> harry-> hermione-> hagrid-> harry-> harry-> hagrid-> ron->
harry-> harry-> hagrid-> hagrid-> hagrid-> hagrid-> hagrid-> hagrid-> harry-> ron-> ron-> ron-> harry-> hermione-> hermione-> h
arry-> harry-> hermione-> hermione-> hagrid-> hermione-> hermione-> ron-> harry-> hermione-> hermione-> harry-> ron-> ron-> hag
rid-> ron-> hermione-> harry-> hermione-> harry-> harry-> hagrid-> hagrid-> harry-> ron-> hermione-> hagrid-> hagrid-> hagrid->
hermione-> ron-> hagrid-> ron-> hagrid-> hagrid-> hagrid-> ron-> hagrid-> harry-> hermione-> harry-> harry-> ron


Random Walk of -> 74 <- state
ron-> harry-> harry-> ron-> hermione-> ron-> hagrid-> ron-> hermione-> ron-> hagrid-> hagrid-> hagrid-> hermione-> harry-> harr
y-> harry-> harry-> hagrid-> hagrid-> ron-> hagrid-> harry-> harry-> harry-> hagrid-> hermione-> ron-> ron-> hagrid-> harry-> h
ermione-> hagrid-> hermione-> hagrid-> ron-> harry-> hagrid-> ron-> harry-> harry-> ron-> harry-> ron-> hermione-> harry-> hagr
id-> harry-> harry-> harry-> hagrid-> harry-> hermione-> ron-> hagrid-> ron-> ron-> hagrid-> ron-> hagrid-> hagrid-> hermione->
ron-> ron-> ron-> hermione-> hagrid-> ron-> hagrid-> hermione-> harry-> hermione-> hagrid-> hermione
```

Figure: 10

The first random walk has 98 states, second one contains 88 states, third one 83 states, fourth one 74 states and the fifth one has 74 states.

Limitation : This project can be done with more than four states, but the limitations of markovchain modules, where this module can plot only four states, that is why we took four states only.

Conclusion: Markov chain is a mathematical system that experiences transitions from one state to another according to certain probabilistic rules. The defining characteristic of a Markov chain is that no matter how the process arrived at its present state, the possible future states are fixed and can be predicted through probabilistic calculations. But there is no surety that prediction will definitely happen.