Lab Code: 20EC605/JO2-B

ARTIFICIAL INTELLIGENCE Lab Manual

Department of Electronics & Communication Engineering

Bapatla Engineering College :: Bapatla

(Autonomous)

G.B.C. Road, Mahatmajipuram, Bapatla-522102, Guntur (Dist.)
Andhra Pradesh, India.
E-Mail:bec.principal@becbapatla.ac.in
Web:www.becbapatla.ac.in

Contents

Bapatla Engineering College :: Bapatla
(Autonomous)

Vision

- To build centers of excellence, impart high quality education and instill high standards of ethics and professionalism through strategic efforts of our dedicated staff, which allows the college to effectively adapt to the ever changing aspects of education.

- To empower the faculty and students with the knowledge, skills and innovative thinking to facilitate discovery in numerous existing and yet to be discovered fields of engineering, technology and interdisciplinary endeavors.

Mission

- Our Mission is to impart the quality education at par with global standards to the students from all over India and in particular those from the local and rural areas.

- We continuously try to maintain high standards so as to make them technologically competent and ethically strong individuals who shall be able to improve the quality of life and economy of our country.

Bapatla Engineering College :: Bapatla

(Autonomous)

Department of Electronics and Communication Engineering

Vision

To produce globally competitive and socially responsible Electronics and Communication Engineering graduates to cater the ever changing needs of the society.

Mission

- To provide quality education in the domain of Electronics and Communication Engineering with advanced pedagogical methods.

- To provide self learning capabilities to enhance employability and entrepreneurial skills and to inculcate human values and ethics to make learners sensitive towards societal issues.

- To excel in the research and development activities related to Electronics and Communication Engineering.

Bapatla Engineering College :: Bapatla

(Autonomous)

Department of Electronics and Communication Engineering

Program Educational Objectives (PEO's)

PEO-I: Equip Graduates with a robust foundation in mathematics, science and Engineering Principles, enabling them to excel in research and higher education in Electronics and Communication Engineering and related fields.

PEO-II: Impart analytic and thinking skills in students to develop initiatives and innovative ideas for Start-ups, Industry and societal requirements.

PEO-III: Instill interpersonal skills, teamwork ability, communication skills, leadership, and a sense of social, ethical, and legal duties in order to promote lifelong learning and Professional growth of the students.

Program Outcomes (PO's)

Engineering Graduates will be able to:

PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7.Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. Individual and Teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Bapatla Engineering College :: Bapatla

(Autonomous)

Department of Electronics and Communication Engineering

Program Specific Outcomes (PSO's)

PSO1: Develop and implement modern Electronic Technologies using analytical methods to meet current as well as future industrial and societal needs.

PSO2: Analyze and develop VLSI, IoT and Embedded Systems for desired specifications to solve real world complex problems.

PSO3: Apply machine learning and deep learning techniques in communication and signal processing.

Artificial Intelligence Lab
III B.Tech. – VI Semester (Code: 20EC605/JO2-B)

Lectures	2	Tutorial	0	Practical	2	Credits	3
Continuous Internal Evaluation			30	Semester End Examination (3 Hours)			70

Prerequisites: None

Course Objectives: Students will be able to

- To Gain a historical perspective of AI and its foundations and to learn the difference between optimal reasoning vs human like reasoning
- To understand the notions of state space representation, exhaustive search, heuristic search along with the time and space complexities and to understand basic principles of AI toward problem solving, inference, perception, knowledge and learning.
- To learn different knowledge representation techniques and to explore the current scope, potential, limitations, and implications of intelligent systems and to explore the current scope, potential, limitations, and implications of intelligent systems.
- To Investigate applications of AI techniques in intelligent agents, expert systems, artificial neural networks and other machine learning models and to understand the applications of AI: namely Game Playing, Theorem Proving, Expert Systems

Course Outcomes: After studying this course, the students will be able to

CO1	Demonstrate the ability to formulate an efficient problem space for a problem.
CO2	Exhibit the ability to select a search algorithm for a problem and characterize its time and space complexities.
CO3	Illustrate the skill of representing knowledge using the appropriate technique.
CO4	Utilize AI techniques to solve problems in Game Playing and Expert Systems.

Mapping of Course Outcomes with Program Outcomes & Program Specific Outcomes

	PO's												**PSO's**		
CO	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**1**	**2**	**3**
CO1	3								3						2
CO2	3	2							3						2
CO3	2	3	2						3						2
CO4	2	3	2		3				3						2
AVG	2.5	2.67	2		3				3						2

LIST OF LAB PROGRAMS

1. Python program to implement Breadth First Search Traversal.

2. Python program to implement Water Jug Problem.

3. Python program to remove punctuations from the given string.

4. Python program to sort the sentence in alphabetical order.

5. Python program to implement Hangman game.

6. Python program to implement Tic-Tac-Toe game.

7. Python program to remove stop words for a given passage from a text file using NLTK.

8. Python program to implement stemming for a given sentence using NLTK.

9. Python program to implement Lemmatization using NLTK.

10. Python program to for Text Classification for the given sentence using NLTK.

1. BREADTH FIRST SEARCH TRAVERSAL

Aim:

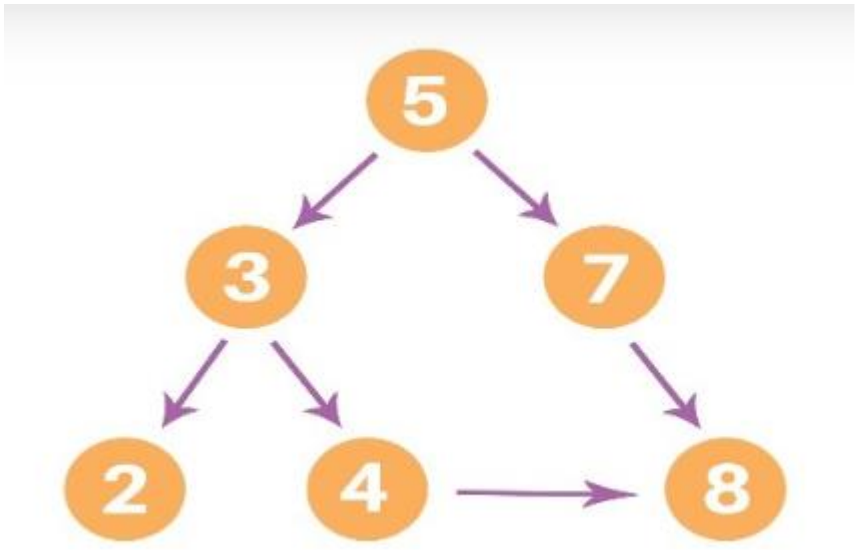
To write a python program to implement Breadth First Search Traversal.

Software Required: IDLE (Python 3.11)

Source Code:

Now, we will see how the source code of the program for implementing breadth first search in python.

Consider the following graph which is implemented in the code below:



```
graph = {
  '5' : ['3','7'],
  '3' : ['2', '4'],
  '7' : ['8'],
  '2' : [],
  '4' : ['8'],
  '8' : []
}

visited = []              # List for visited nodes.

queue = []                #Initialize a queue
```

```
def bfs(visited, graph, node):        #function for BFS
  visited.append(node)

  queue.append(node)

  while queue:                      # Creating loop to visit each node

    m = queue.pop(0)

    print (m, end = " ")

    for neighbour in graph[m]:

      if neighbour not in visited:

        visited.append(neighbour)

        queue.append(neighbour)

# Driver Code

print("Following is the Breadth-First Search")

bfs(visited, graph, '5')                    # function calling
```

<u>Output:</u>

```
Following is the Breadth-First Search
5 3 7 2 4 8
```

=== Code Execution Successful ===

2. WATER JUG PROBLEM

Aim:

To write a python program to implement Water Jug Problem.

Software Required: IDLE (Python 3.11)

Source Code:

```
# This function is used to initialize the
# dictionary elements with a default value.

from collections import defaultdict

# jug1 and jug2 contain the value
# for max capacity in respective jugs
# and aim is the amount of water to be measured.

jug1, jug2, aim = 4, 3, 2

# Initialize dictionary with
# default value as false.

visited = defaultdict(lambda: False)

# Recursive function which prints the
# intermediate steps to reach the final
# solution and return boolean value
# (True if solution is possible, otherwise False).
# amt1 and amt2 are the amount of water present
# in both jugs at a certain point of time.

def waterJugSolver(amt1, amt2):

    # Checks for our goal and
    # returns true if achieved.

    if (amt1 == aim and amt2 == 0) or (amt2 == aim and amt1 == 0):
        print(amt1, amt2)
        return True

    # Checks if we have already visited the
    # combination or not. If not, then it proceeds further.

    if visited[(amt1, amt2)] == False:
        print(amt1, amt2)

        # Changes the boolean value of
        # the combination as it is visited.

        visited[(amt1, amt2)] = True

        # Check for all the 6 possibilities and
        # see if a solution is found in any one of them.

        return (waterJugSolver(0, amt2) or
                waterJugSolver(amt1, 0) or
                waterJugSolver(jug1, amt2) or
                waterJugSolver(amt1, jug2) or
                waterJugSolver(amt1 + min(amt2, (jug1-amt1)),
                amt2 - min(amt2, (jug1-amt1))) or
                waterJugSolver(amt1 - min(amt1, (jug2-amt2)),
                amt2 + min(amt1, (jug2-amt2))))

    # Return False if the combination is
    # already visited to avoid repetition otherwise
    # recursion will enter an infinite loop.

    else:
        return False

print("Steps: ")

# Call the function and pass the
# initial amount of water present in both jugs.

waterJugSolver(0, 0)
```

Output:

```
Steps:
0 0
4 0
4 3
0 3
3 0
3 3
4 2
0 2

=== Code Execution Successful ===
```

3. REMOVE PUNCTUATIONS FROM THE GIVEN STRING

Aim:

To write a python program to remove punctuations from the given string.

Software Required: IDLE (Python 3.11)

Source Code:

```
# define punctuation

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

#my_str = "Hello!!!, he said ---and went."
# To take input from the user

my_str = input("Enter a string: ")

# remove punctuation from the string

no_punct = ""
for char in my_str:
   if char not in punctuations:
       no_punct = no_punct + char

# display the unpunctuated string

print(no_punct)
```

Output:

```
Enter a string: Hello!!!, he said ---and went.
Hello he said and went

=== Code Execution Successful ===
```

4. SORT THE SENTENCE IN ALPHABETICAL ORDER

Aim:

To write a python program to sort the sentence in alphabetical order.

Software Required: IDLE (Python 3.11)

Source Code:

```
# The following code sorts words in alphabetical order
# Taking user input

sentence = input("Enter a sentence: ")

# Splitting the sentence into words

words = sentence.split()

# Sorting the words

words.sort()

# Printing the sorted words

print("The sorted words are:")
for word in words:
 print(word)
```

Output:

```
Enter a sentence: I am studying in Bapatla Engineering College
The sorted words are:
Bapatla
College
Engineering
I
am
in
studying

=== Code Execution Successful ===
```

5. HANGMAN GAME

Aim:

To write a python program to implement Hangman game.

Software Required: IDLE (Python 3.11)

Source Code:

```
#importing the time module

import time

#welcoming the user

name = input("What is your name? ")
print ("Hello, " + name, "Time to play hangman!")

#wait for 1 second

time.sleep(1)
print ("Start guessing...")
time.sleep(0.5)

#here we set the secret. You can select any word to play with.
word = ("secret")

#creates an variable with an empty value
guesses = ''

#determine the number of turns
turns = 10

# Create a while loop
#check if the turns are more than zero

while turns > 0:
```

```
# make a counter that starts with zero

failed = 0

# for every character in secret_word

for char in word:

# see if the character is in the players guess

    if char in guesses:

    # print then out the character

        print (char,end=""),
    else:

    # if not found, print a dash

        print ("_",end=""),

    # and increase the failed counter with one

        failed += 1

# if failed is equal to zero
# print You Won

if failed == 0:
    print ("You won")

# exit the script

    break

# ask the user go guess a character

guess = input("guess a character:")

# set the players guess to guesses
```

```
guesses += guess

# if the guess is not found in the secret word

if guess not in word:

 # turns counter decreases with 1 (now 9)

   turns -= 1

# print wrong

   print ("Wrong")

# how many turns are left

   print ("You have", + turns, 'more guesses' )

# if the turns are equal to zero

   if turns == 0:

   # print "You Lose"

      print ("You Lose"  )
```

Output:

```
What is your name? sumanth
Hello, sumanth Time to play hangman!
Start guessing...
______guess a character:h
Wrong
You have 9 more guesses
______guess a character:e
_e__e_guess a character:s
se__e_guess a character:c
sec_e_guess a character:r
secre_guess a character:t
secret You won

=== Code Execution Successful ===
```

6. TIC-TAC-TOE GAME

Aim:

To write a python program to implement Tic-Tac-Toe game.

Software Required: IDLE (Python 3.11)

Source Code:

```
# Tic-Tac-Toe Program using random number in Python
# importing all necessary libraries

import numpy as np
import random
from time import sleep

# Creates an empty board

def create_board():
    return(np.array([[0, 0, 0],
                     [0, 0, 0],
                     [0, 0, 0]]))

# Check for empty places on board

def possibilities(board):
    l = []

    for i in range(len(board)):
        for j in range(len(board)):

            if board[i][j] == 0:
                l.append((i, j))
    return(l)

# Select a random place for the player

def random_place(board, player):
    selection = possibilities(board)
    current_loc = random.choice(selection)
    board[current_loc] = player
    return(board)

# Checks whether the player has three
# of their marks in a horizontal row

def row_win(board, player):
    for x in range(len(board)):
        win = True

        for y in range(len(board)):
            if board[x, y] != player:
                win = False
                continue

        if win == True:
            return(win)
    return(win)

# Checks whether the player has three
# of their marks in a vertical row

def col_win(board, player):
    for x in range(len(board)):
        win = True

        for y in range(len(board)):
            if board[y][x] != player:
                win = False
                continue

        if win == True:
            return(win)
    return(win)

# Checks whether the player has three
# of their marks in a diagonal row
```

```
def diag_win(board, player):
    win = True
    y = 0
    for x in range(len(board)):
        if board[x, x] != player:
            win = False
    if win:
        return win
    win = True
    if win:
        for x in range(len(board)):
            y = len(board) - 1 - x
            if board[x, y] != player:
                win = False
    return win
```

# Evaluates whether there is
# a winner or a tie

```
def evaluate(board):
    winner = 0

    for player in [1, 2]:
        if (row_win(board, player) or
                col_win(board, player) or
                diag_win(board, player)):

            winner = player

    if np.all(board != 0) and winner == 0:
        winner = -1
    return winner
```

# Main function to start the game

```
def play_game():
    board, winner, counter = create_board(), 0, 1
    print(board)
    sleep(2)
    while winner == 0:
        for player in [1, 2]:
            board = random_place(board, player)
            print("Board after " + str(counter) + " move")
            print(board)
            sleep(2)
            counter += 1
            winner = evaluate(board)
            if winner != 0:
                break
    return(winner)
```

# Driver Code

```
print("Winner is: " + str(play_game()))
```

Output:

```
 [[0 0 0]
 [0 0 0]
 [0 0 0]]
Board after 1 move
[[0 0 0]
 [0 0 0]
 [1 0 0]]
Board after 2 move
[[0 0 0]
 [0 2 0]
 [1 0 0]]
Board after 3 move
[[0 1 0]
 [0 2 0]
 [1 0 0]]
Board after 4 move
[[0 1 0]
 [2 2 0]
 [1 0 0]]
Board after 5 move
[[1 1 0]
```

```
 [2 2 0]
 [1 0 0]]
Board after 6 move
[[1 1 0]
 [2 2 0]
 [1 2 0]]
Board after 7 move
[[1 1 0]
 [2 2 0]
 [1 2 1]]
Board after 8 move
[[1 1 0]
 [2 2 2]
 [1 2 1]]
Winner is: 2
```

=== Code Execution Successful ===

7. REMOVE STOP WORDS FOR A GIVEN PASSAGE FROM A TEXT FILE USING NLTK

Aim:

To write a python program to remove stop words for a given passage from a text file using NLTK.

Software Required: IDLE (Python 3.11)

Source Code:

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')

def remove_stop_words(string):
    stop_words = set(stopwords.words('english'))
    words = string.split()
    filtered_words = [word for word in words if word.lower() not in
stop_words]
    new_string = ' '.join(filtered_words)
    return new_string

# Example usage

input_string = "This is an example sentence to remove stop words from."
result = remove_stop_words(input_string)
print("Original string:", input_string)
print("Modified string:", result)
```

Output:

```
Original string: This is an example sentence to remove stop words from.
Modified string: example sentence remove stop words from.

=== Code Execution Successful ===
```

8. STEMMING FOR A GIVEN SENTENCE USING NLTK

Aim:

To write a python program to implement stemming for a given sentence using NLTK.

Software Required: IDLE (Python 3.11)

Source Code:

```
# import these modules
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

ps = PorterStemmer()

# choose some words to be stemmed

words = ["program", "programs", "programmer", "programming",
"programmers"]

for w in words:
    print(w, " : ", ps.stem(w))
```

Output:

```
program  :  program
programs  :  program
programmer  :  programm
programming  :  program
programmers  :  program

=== Code Execution Successful ===
```

9. LEMMATIZATION USING NLTK

Aim:

To write a python program to implement Lemmatization using NLTK.

Software Required: IDLE (Python 3.11)

Source Code:

```
# import these modules

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print("rocks :", lemmatizer.lemmatize("rocks"))
print("corpora :", lemmatizer.lemmatize("corpora"))

# a denotes adjective in "pos"

print("better :", lemmatizer.lemmatize("better", pos="a"))
```

Output:

```
rocks : rock
corpora : corpus
better : good
```

=== Code Execution Successful ===

10. TEXT CLASSIFICATION FOR THE GIVEN SENTENCE USING NLTK.

Aim:

To write a python program for Text Classification for the given sentence using NLTK.

Software Required: IDLE (Python 3.11)

Source Code:

```
import nltk
import random
# Import the movie_reviews corpus
from nltk.corpus import movie_reviews

# Download necessary resources
nltk.download('movie_reviews')

# Load the movie reviews dataset
documents = [(list(movie_reviews.words(fileid)), category)
             for category in movie_reviews.categories()
             for fileid in movie_reviews.fileids(category)]
random.shuffle(documents)

# Create a list of all words in the dataset
all_words = nltk.FreqDist(w.lower() for w in movie_reviews.words())

# Select the top 2000 words as features
word_features = list(all_words)[:2000]
```

```
def document_features(document):
    """Extract features from a document."""
    document_words = set(document)
    features = {}
    for word in word_features:
        features[f'contains({word})'] = (word in document_words)
    return features

# Create feature sets
featuresets = [(document_features(doc), category) for (doc, category) in documents]

# Split data into training and testing sets
train_set, test_set = featuresets[1000:], featuresets[:1000]

# Train a Naive Bayes classifier
classifier = nltk.NaiveBayesClassifier.train(train_set)

# Evaluate the classifier
print(f'Accuracy: {nltk.classify.accuracy(classifier, test_set)}')
classifier.show_most_informative_features(10)
```

Output:

```
[nltk_data] Downloading package movie_reviews to /root/nltk_data...
[nltk_data]   Package movie_reviews is already up-to-date!
Accuracy: 0.816
Most Informative Features
contains(outstanding) = True          pos : neg    =     15.6 : 1.0
contains(damon) = True                pos : neg    =     11.9 : 1.0
contains(wonderfully) = True          pos : neg    =      7.1 : 1.0
contains(remake) = True               neg : pos    =      7.1 : 1.0
contains(seagal) = True               neg : pos    =      6.9 : 1.0
contains(poorly) = True               neg : pos    =      6.4 : 1.0
contains(awful) = True                neg : pos    =      6.0 : 1.0
contains(era) = True                  pos : neg    =      5.8 : 1.0
contains(wasted) = True               neg : pos    =      5.7 : 1.0
contains(portrayal) = True            pos : neg    =      5.5 : 1.0
```

=== Code Execution Successful ===

Appendix

Installing NLTK Data

NLTK comes with many corpora, toy grammars, trained models, etc. A complete list is posted at: https://www.nltk.org/nltk_data/

To install the data, first install NLTK (see https://www.nltk.org/install.html), then use NLTK's data downloader as described below.

Apart from individual data packages, you can download the entire collection (using "all"), or just the data required for the examples and exercises in the book (using "book"), or just the corpora and no grammars or trained models (using "all-corpora").

Interactive installer

For central installation on a multi-user machine, do the following from an administrator account.

Run the Python interpreter and type the commands:

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

A new window should open, showing the NLTK Downloader. Click on the File menu and select Change Download Directory. For central installation, set this to C:\nltk_data (Windows), /usr/local/share/nltk_data (Mac), or /usr/share/nltk_data (Unix). Next, select the packages or collections you want to download.

If you did not install the data to one of the above central locations, you will need to set the NLTK_DATA environment variable to specify the location of the data. (On a Windows machine, right click on “My Computer” then select Properties > Advanced > Environment Variables > User Variables > New...)

Test that the data has been installed as follows. (This assumes you downloaded the Brown Corpus):

```
>>> from nltk.corpus import brown
>>> brown.words()
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
```

Installing via a proxy web server

If your web connection uses a proxy server, you should specify the proxy address as follows. In the case of an authenticating proxy, specify a username and password. If the proxy is set to None then this function will attempt to detect the system proxy.

```
>>> nltk.set_proxy('http://proxy.example.com:3128', ('USERNAME', 'PASSWORD'))
>>> nltk.download()
```

Command line installation

The downloader will search for an existing nltk_data directory to install NLTK data. If one does not exist it will attempt to create one in a central location (when using an administrator account) or otherwise in the user’s filespace. If necessary, run the download command from an administrator account, or using sudo. The recommended system location is C:\nltk_data (Windows); /usr/local/share/nltk_data (Mac); and /usr/share/nltk_data (Unix). You can use the -d flag to specify a different location (but if you do this, be sure to set the NLTK_DATA environment variable accordingly).

Run the command python -m nltk.downloader all. To ensure central installation, run the command sudo python -m nltk.downloader -d /usr/local/share/nltk_data all.

Windows: Use the “Run...” option on the Start menu. Windows Vista users need to first turn on this option, using Start -> Properties -> Customize to check the box to activate the “Run...” option.

Test the installation: Check that the user environment and privileges are set correctly by logging in to a user account, starting the Python interpreter, and accessing the Brown Corpus (see the previous section).

Manual installation

Create a folder nltk_data, e.g. C:\nltk_data, or /usr/local/share/nltk_data, and subfolders chunkers, grammars, misc, sentiment, taggers, corpora, help, models, stemmers, tokenizers.

Download individual packages from https://www.nltk.org/nltk_data/ (see the “download” links). Unzip them to the appropriate subfolder. For example, the Brown Corpus, found at: https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/packages/corpora/brown.zip is to be unzipped to nltk_data/corpora/brown.

Set your NLTK_DATA environment variable to point to your top level nltk_data folder.

References

1. Artificial Intelligence: Building Intelligent Systems ByParag Kulkarni and Prachi Joshi, PHI Publications.
2. Russell, Norvig: Artificial intelligence, A Modern Approach, Pearson Education, Second Edition. 2004.
3. Rich, Knight, Nair: Artificial intelligence, Tata McGraw Hill, Third Edition 2009.
4. Introduction to Artificial Intelligence by Eugene Charniak, Pearson.
5. Introduction to Artificial Intelligence and expert systems Dan W.Patterson. PHI.
6. Artificial Intelligence by George Flugerrearson fifth edition.
7. Saroj Kaushik. Artificial Intelligence. Cengage Learning. 2011