```
# Implementation of Tokenization, Lemmatization, Stemming and stopwords
import nltk
nltk.download('punkt_tab')
data = "hello world i am going outside for playing cricket"
sent_tokens = nltk.sent_tokenize(data)
print(sent_tokens)
word_tokens = nltk.word_tokenize(data)
print(word_tokens)
#stemming
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
for word in word_tokens:
  print(stemmer.stem(word))
#lemmatization
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
lemmatizer = WordNetLemmatizer()
for word in word_tokens:
  print(lemmatizer.lemmatize(word))
#stopwords
from nltk.corpus import stopwords
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
for word in word_tokens:
  if word not in stop_words:
    print(word)
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
['hello world i am going outside for playing cricket']
['hello', 'world', 'i', 'am', 'going', 'outside', 'for', 'playing', 'cricket']
hello
world
i
am
go
outsid
for
play
cricket
hello
world
i
am
going
outside
for
playing
cricket
hello
world
going
outside
playing
cricket
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
# Implement Custom Spell Checker
!pip install spello
from spello.model import SpellCorrectionModel
model = SpellCorrectionModel(language='en')
with open ('word.txt', 'r') as file:
  data = file.readlines()
  for words in data:
    model.train([words.strip()])
```

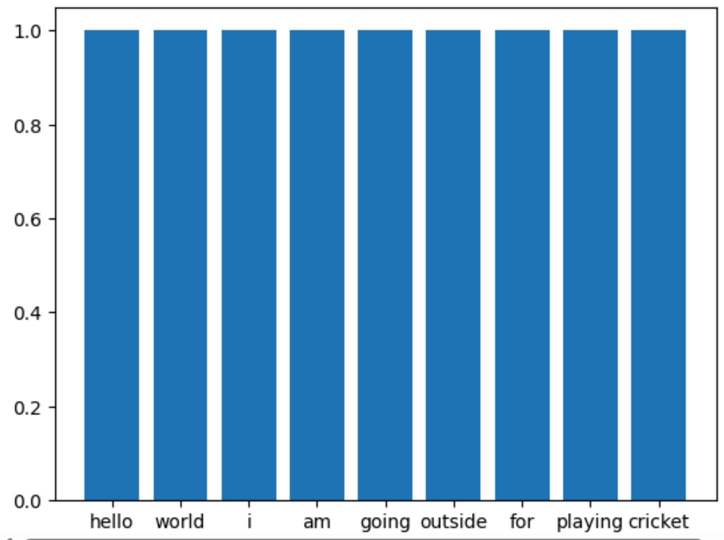
```
Requirement already satisfied: spello in /usr/local/lib/python3.11/dist-packages (1.3.0)
Requirement already satisfied: nltk<4,>=3.4.5 in /usr/local/lib/python3.11/dist-packages (from spello) (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nltk<4,>=3.4.5->spello) (8.1.8)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from nltk<4,>=3.4.5->spello) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/dist-packages (from nltk<4,>=3.4.5->spello) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from nltk<4,>=3.4.5->spello) (4.67.1)
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-5-c249a134ebef> in <cell line: 0>()
      3 from spello.model import SpellCorrectionModel
      4 model = SpellCorrectionModel(language='en')
----> 5 with open ('word.txt', 'r') as file:
      6   data = file.readlines()
      7   for words in data:

FileNotFoundError: [Errno 2] No such file or directory: 'word.txt'
```

Next steps: Explain error

```
# Implement Word Segmentation and Sentence Segmentation
!pip install spacy
import spacy
spacy.cli.download("en_core_web_sm")
nlp = spacy.load('en_core_web_sm')
doc = nlp("hello world i am going outside for playing cricket")
for token in doc.sents:
  print(token.text)
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
text = "hello world i am going outside for playing cricket"
words = word_tokenize(text)
word_freq = FreqDist(words)
print(word_freq)
print("Frequency of word is in text is", word_freq.freq('is'))
import matplotlib.pyplot as plt
plt.bar(word_freq.keys(), word_freq.values())
plt.show()
```

```
Requirement already satisfied: spacy in /usr/local/lib/python3.11/dist-packages (3.8.4)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (1.0.12)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.0.11)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.0.9)
Requirement already satisfied: thinc<8.4.0,>=8.3.4 in /usr/local/lib/python3.11/dist-packages (from spacy) (8.3.4)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.11/dist-packages (from spacy) (1.1.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.5.1)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.0.10)
Requirement already satisfied: weasel<0.5.0,>=0.1.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (0.4.1)
Requirement already satisfied: typer<1.0.0,>=0.3.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (0.15.2)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (4.67.1)
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.0.2)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.32.3)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.11/dist-packages (from spacy) (2.10.6)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.1.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from spacy) (75.1.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (24.2)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.11/dist-packages (from spacy) (3.5.0)
Requirement already satisfied: language-data>=1.2 in /usr/local/lib/python3.11/dist-packages (from langcodes<4.0.0,>=3.2.0->spacy) (1
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>
Requirement already satisfied: pydantic-core==2.27.2 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=
Requirement already satisfied: typing-extensions>=4.12.2 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0->spa
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2
Requirement already satisfied: blis<1.3.0,>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from thinc<8.4.0,>=8.3.4->spacy) (1.2.0
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.11/dist-packages (from thinc<8.4.0,>=8.3.4->spacy)
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->spacy) (8.1.8)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->spacy) (1.5.4
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->spacy) (13.9.4)
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from weasel<0.5.0,>=0.1.0->spac
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/python3.11/dist-packages (from weasel<0.5.0,>=0.1.0->spacy)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->spacy) (3.0.2)
Requirement already satisfied: marisa-trie>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from language-data>=1.2->langcodes<4.0.
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0.0,>=0.
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0.0,>=
Requirement already satisfied: wrapt in /usr/local/lib/python3.11/dist-packages (from smart-open<8.0.0,>=5.2.1->weasel<0.5.0,>=0.1.0-
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich>=10.11.0->type
✓ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
⚠ Restart to reload dependencies
If you are in a Jupyter or Colab notebook, you may need to restart Python in
order to load all the package's dependencies. You can do this by selecting the
'Restart kernel' or 'Restart runtime' option.
hello world i am going outside for playing cricket
<FreqDist with 9 samples and 9 outcomes>
Frequency of word is in text is 0.0
```



```
# Implementation of CountVectorizer
from sklearn.feature_extraction.text import CountVectorizer
corpus  = ['this is RVRJC', 'This is Guntur', 'This is Warangal']
print(corpus)
vect = CountVectorizer()
vect.fit(corpus)
x = vect.fit_transform(corpus)
```

```
print(x)
text = ['This is My world !!!']
bi_gram = CountVectorizer(ngram_range = (2, 2))
print(bi_gram.fit_transform(text))
print(bi_gram.vocabulary_)
text = ['This is my New World !!']
tri_gram = CountVectorizer(ngram_range = (3, 3))
print(tri_gram.fit_transform(text))
print(tri_gram.vocabulary_)
text = ["Hello World !!, This is Chief...."]
tri_gram = CountVectorizer(ngram_range = (1, 3))
print(tri_gram.fit_transform(text))
print(tri_gram.vocabulary_)
```

```
['this is RVRJC', 'This is Guntur', 'This is Warangal']
<Compressed Sparse Row sparse matrix of dtype 'int64'
	with 9 stored elements and shape (3, 5)>
  Coords	Values
  (0, 3)	1
  (0, 1)	1
  (0, 2)	1
  (1, 3)	1
  (1, 1)	1
  (1, 0)	1
  (2, 3)	1
  (2, 1)	1
  (2, 4)	1
<Compressed Sparse Row sparse matrix of dtype 'int64'
	with 3 stored elements and shape (1, 3)>
  Coords	Values
  (0, 2)	1
  (0, 0)	1
  (0, 1)	1
{'this is': 2, 'is my': 0, 'my world': 1}
<Compressed Sparse Row sparse matrix of dtype 'int64'
	with 3 stored elements and shape (1, 3)>
  Coords	Values
  (0, 2)	1
  (0, 0)	1
  (0, 1)	1
{'this is my': 2, 'is my new': 0, 'my new world': 1}
<Compressed Sparse Row sparse matrix of dtype 'int64'
	with 12 stored elements and shape (1, 12)>
  Coords	Values
  (0, 1)	1
  (0, 9)	1
  (0, 6)	1
  (0, 4)	1
  (0, 0)	1
  (0, 2)	1
  (0, 10)	1
  (0, 7)	1
  (0, 5)	1
  (0, 3)	1
  (0, 11)	1
  (0, 8)	1
{'hello': 1, 'world': 9, 'this': 6, 'is': 4, 'chief': 0, 'hello world': 2, 'world this': 10, 'this is': 7, 'is chief': 5, 'hello world t
```

```
# Implementing Bag of Words and PoS tagging
from sklearn.feature_extraction.text import CountVectorizer
text = ["hello world i am going outside for playing cricket"]
vect = CountVectorizer()
vect.fit(text)
x = vect.fit_transform(text)
print(x)
print(vect.vocabulary_)
t = ["I love NLP"]
vect.fit(t)
y = vect.fit_transform(t)
print(y)
print(vect.vocabulary_)
x = vect.fit_transform(text + t)
print(vect.transform(t).toarray())
print(vect.vocabulary_)
import spacy
nlp = spacy.load('en_core_web_sm')
doc = nlp("hello world i am going outside for playing cricket")
for token in doc:
  print(token.text, token.pos_)
```

```
<Compressed Sparse Row sparse matrix of dtype 'int64'
        with 8 stored elements and shape (1, 8)>
  Coords        Values
  (0, 4)        1
  (0, 7)        1
  (0, 0)        1
  (0, 3)        1
  (0, 5)        1
  (0, 2)        1
  (0, 6)        1
  (0, 1)        1
{'hello': 4, 'world': 7, 'am': 0, 'going': 3, 'outside': 5, 'for': 2, 'playing': 6, 'cricket': 1}
<Compressed Sparse Row sparse matrix of dtype 'int64'
        with 2 stored elements and shape (1, 2)>
  Coords        Values
  (0, 0)        1
  (0, 1)        1
{'love': 0, 'nlp': 1}
[[0 0 0 0 0 1 1 0 0 0]]
{'hello': 4, 'world': 9, 'am': 0, 'going': 3, 'outside': 7, 'for': 2, 'playing': 8, 'cricket': 1, 'love': 5, 'nlp': 6}
hello INTJ
world NOUN
i PRON
am AUX
going VERB
outside ADV
for ADP
playing VERB
cricket NOUN
```

```
# Implementation of Multiword Expression
import nltk
from nltk.tokenize import MWETokenizer, word_tokenize
m = MWETokenizer([('Hong', 'Kong'), ('takes', 'off')], separator = ' - ')
t1 = "The City in China is Hong Kong"
t2 = "The Fligh takes off from Hong Kong to India"
print(m.tokenize(word_tokenize(t1)))
print(m.tokenize(word_tokenize(t2)))
```

```
['The', 'City', 'in', 'China', 'is', 'Hong - Kong']
['The', 'Fligh', 'takes - off', 'from', 'Hong - Kong', 'to', 'India']
```

```
# Implementation of Cleaning Functions
!pip install clean_text
from cleantext import clean
s1 = 'CSE-DS'
print(clean(s1, fix_unicode = True))
s2 = 'k\0047\ttg\eccv\3232'
print(clean(s2, to_ascii = True))
```

```
Requirement already satisfied: clean_text in /usr/local/lib/python3.11/dist-packages (0.6.0)
Requirement already satisfied: emoji<2.0.0,>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from clean_text) (1.7.0)
Requirement already satisfied: ftfy<7.0,>=6.0 in /usr/local/lib/python3.11/dist-packages (from clean_text) (6.3.1)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.11/dist-packages (from ftfy<7.0,>=6.0->clean_text) (0.2.13)
cse-ds
k7 tg\eccvo2
```

```
# Implement TF-IDF
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
vect = TfidfVectorizer()
text = ["hello world i am going outside for playing cricket"]
vect.fit(text)
x = vect.fit_transform(text)
print(x)
print(vect.vocabulary_)
print(vect.get_feature_names_out)
matrix = pd.DataFrame(x.toarray(), columns = vect.get_feature_names_out())
matrix
```

```
<Compressed Sparse Row sparse matrix of dtype 'float64'
	with 8 stored elements and shape (1, 8)>
  Coords	Values
  (0, 4)	0.35355339059327373
  (0, 7)	0.35355339059327373
  (0, 0)	0.35355339059327373
  (0, 3)	0.35355339059327373
  (0, 5)	0.35355339059327373
  (0, 2)	0.35355339059327373
  (0, 6)	0.35355339059327373
  (0, 1)	0.35355339059327373
{'hello': 4, 'world': 7, 'am': 0, 'going': 3, 'outside': 5, 'for': 2, 'playing': 6, 'cricket': 1}
<bound method CountVectorizer.get_feature_names_out of TfidfVectorizer()>
```

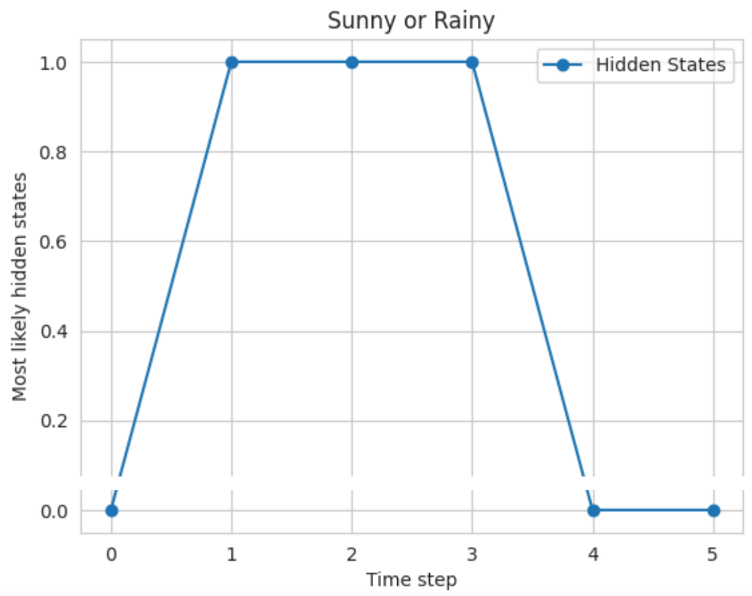
	am	cricket	for	going	hello	outside	playing	world
0	0.353553	0.353553	0.353553	0.353553	0.353553	0.353553	0.353553	0.353553

```
# Implement WOrdNet and Word Sense Disambiguation
import nltk
nltk.download('wordnet')
from nltk.tokenize import word_tokenize
from nltk.wsd import lesk
a1 = lesk(word_tokenize("There is a Traffic Jam"), "jam")
a2 = lesk(word_tokenize("This is a Network delay due to Network Jam"), 'jam')
print(a1, a1.definition())
print(a2, a2.definition())
from nltk.corpus import wordnet
print(wordnet.synsets('jam'))
synsets = wordnet.synsets('Hello')
for synset in synsets:
  print(synset.definition())
  print(synset.examples())
  print(synset.name())
  print(synset.pos())
```

```
Synset('fix.n.01') informal terms for a difficult situation
Synset('jam.v.06') crowd or pack to capacity
[Synset('jam.n.01'), Synset('fix.n.01'), Synset('crush.n.02'), Synset('jamming.n.01'), Synset('throng.v.01'), Synset('jam.v.02'), Synset
an expression of greeting
['every morning they exchanged polite hellos']
hello.n.01
n
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
!pip install hmmlearn
import hmmlearn
from hmmlearn import hmm
import numpy as np
from hmmlearn.hmm import CategoricalHMM
import seaborn as sns
import matplotlib.pyplot as plt
states = ['Sunny', 'Rainy']
n_states = len(states)
observations = ['Dry', 'wet']
n_observations = len(observations)
start_probability = np.array([0.6, 0.4])
trans_matrix = np.array([[0.7, 0.3],
                        [0.3, 0.7]])
emission_matrix = np.array([[0.9, 0.1],
                           [0.2, 0.8]])
model = hmm.CategoricalHMM(n_components=n_states)
model.startprob_=start_probability
model.transmat_=trans_matrix
model.emissionprob_=emission_matrix
observation_seq=np.array([0,1,0,1,0,0]).reshape(-1,1)
observation_seq
hidden_states = model.predict(observation_seq)
print(hidden_states)
logprob, hidden_states = model.decode(observation_seq,lengths=len(observation_seq), algorithm="viterbi"
sns.set_style('whitegrid')
plt.plot(hidden_states, marker='o', label="Hidden States")
plt.xlabel('Time step')
plt.ylabel('Most likely hidden states')
plt.title("Sunny or Rainy")
plt.legend()
```

```
Collecting hmmlearn
  Downloading hmmlearn-0.3.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3 0 kB)
Requirement already satisfied: numpy>=1.10 in /usr/local/lib/python3.11/dist-packages (from hmmlear ) (2.0.2)
Requirement already satisfied: scikit-learn!=0.22.0,>=0.16 in /usr/local/lib/python3.11/dist-packag s (from hmmlearn) (1.6.1)
Requirement already satisfied: scipy>=0.19 in /usr/local/lib/python3.11/dist-packages (from hmmlear ) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from sciki -learn!=0.22.0,>=0.16->hmmlearn) (1.
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (frc  scikit-learn!=0.22.0,>=0.16->hmmlea
Downloading hmmlearn-0.3.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (165 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 165.9/165.9 kB 3.0 MB/s eta 0:00:00
Installing collected packages: hmmlearn
Successfully installed hmmlearn-0.3.3
[0 1 1 1 0 0]
```



```
def viterbi(obs, states, start_p, trans_p, emit_p):
    V = [{}]
    for st in states:
        V[0][st] = {"prob": start_p[st] * emit_p[st][obs[0]], "prev": None}
    for t in range(1, len(obs)):
        V.append({})
        for st in states:
            max_tr_prob = max(
                V[t - 1][prev_st]["prob"] * trans_p[prev_st][st] for prev_st in states
            )
            for prev_st in states:
                if V[t - 1][prev_st]["prob"] * trans_p[prev_st][st] == max_tr_prob:
                    max_prob = max_tr_prob * emit_p[st][obs[t]]
                    V[t][st] = {"prob": max_prob, "prev": prev_st}
                    break
    return V
states = ("Rainy", "Sunny")
observations = ("walk", "shop", "clean")
start_prob = {"Rainy": 0.6, "Sunny":0.4}
transition_prob={'Rainy':{'Rainy':0.7,'Sunny':0.3},'Sunny':{'Rainy':0.4,'Sunny':0.6}}
emission_prob={'Rainy':{'walk':0.1,'shop':0.4,'clean':0.5},'Sunny':{'walk':0.6,'shop':0.3,'clean':0.1}}
viterbi(observations,states,start_prob,transition_prob,emission_prob)
```

```
[{'Rainy': {'prob': 0.06, 'prev': None},
  'Sunny': {'prob': 0.24, 'prev': None}},
 {'Rainy': {'prob': 0.038400000000000004, 'prev': 'Sunny'},
  'Sunny': {'prob': 0.043199999999999995, 'prev': 'Sunny'}},
 {'Rainy': {'prob': 0.01344, 'prev': 'Rainy'},
  'Sunny': {'prob': 0.0025919999999999997, 'prev': 'Sunny'}}]
```