## Import Library

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn import preprocessing
from sklearn import model_selection
```

## Load Dataset & Pre Processing

In [2]:
```
def process_tag(tag):
    tag = [t.split("-")[1] if t != 'O' else t for t in tag]
    tag = ["<s>"] + tag
    return tag

def process_sentence(sentence):
    sentence = [sent.lower() for sent in sentence]
    sentence = ["<s>"] + sentence # Add Start Token
    return sentence

def process_data(data_path):
    df = pd.read_csv(data_path, encoding="latin-1")
    df.loc[:, "Sentence #"] = df["Sentence #"].fillna(method="ffill")

    sentences = df.groupby("Sentence #")["Word"].apply(list).values
    tags = df.groupby("Sentence #")["Tag"].apply(list).values

    sentences = [process_sentence(sentence) for sentence in sentences]
    tags = [process_tag(tag) for tag in tags]

    return sentences, tags

data_path = r"Keerthana\Dataset\NER  Dataset\ner_dataset.csv"
sentences, tags = process_data(data_path)
```

## Hidden Markov Model

In [3]:
```
# No of Tags
# <s> - Start Token

tag = set()
for t in tags:
    tag.update(set(t))

print("Number of Tags :", len(tag))
print("Tags :", list(tag))
```

```
Number of Tags : 10
Tags : ['art', 'per', 'gpe', 'eve', 'nat', 'org', 'tim', 'geo', '<s>', 'O']
```

## Transition Matrix

In [4]:
```
def create_transition_matrix(tags):

    # Bigram
    bigram = {}
    for tag in tags:
        for idx in range(len(tag)-1):
            b_tuple = (tag[idx], tag[idx+1])
            if(bigram.get(b_tuple, -1) == -1):
                bigram[b_tuple] = 1
            else:
                bigram[b_tuple] += 1

    # Tags
    tag = set()
    for t in tags:
        tag.update(set(t))
    no_tag = len(tag)
    tag = list(tag)

    # Transition Matrix
    transition_matrix = pd.DataFrame(np.zeros((no_tag, no_tag)), index=tag, columns=tag)

    # Populate Transition Matrix
    for tag_first in tag:
        for tag_second in tag:
            transition_matrix[tag_first][tag_second] = bigram.get((tag_first, tag_second), 0)

    transition_matrix = transition_matrix / transition_matrix.sum(axis=0)

    return transition_matrix

transition_matrix = create_transition_matrix(tags)
transition_matrix.T
```

Out[4]:

| | art | per | gpe | eve | nat | org | tim | geo | <s> | O |
|---|---|---|---|---|---|---|---|---|---|---|
| art | 0.425501 | 0.005731 | 0.000000 | 0.000000 | 0.000000 | 0.004298 | 0.012894 | 0.002865 | 0.0 | 0.548711 |
| per | 0.000000 | 0.504017 | 0.001870 | 0.000000 | 0.000058 | 0.011132 | 0.005843 | 0.006895 | 0.0 | 0.470184 |
| gpe | 0.000062 | 0.081601 | 0.012324 | 0.000062 | 0.000000 | 0.026516 | 0.001930 | 0.008465 | 0.0 | 0.869040 |
| eve | 0.000000 | 0.000000 | 0.000000 | 0.451786 | 0.000000 | 0.000000 | 0.010714 | 0.000000 | 0.0 | 0.537500 |
| nat | 0.000000 | 0.007937 | 0.000000 | 0.000000 | 0.202381 | 0.000000 | 0.003968 | 0.003968 | 0.0 | 0.781746 |
| org | 0.000217 | 0.017661 | 0.004036 | 0.000054 | 0.000000 | 0.454641 | 0.008776 | 0.001192 | 0.0 | 0.513422 |
| tim | 0.000149 | 0.001899 | 0.002123 | 0.000819 | 0.000037 | 0.001937 | 0.243128 | 0.004655 | 0.0 | 0.745251 |
| geo | 0.000089 | 0.002888 | 0.003243 | 0.000022 | 0.000000 | 0.001533 | 0.020525 | 0.164686 | 0.0 | 0.807015 |
| <s> | 0.000375 | 0.083801 | 0.062324 | 0.000209 | 0.000229 | 0.057382 | 0.010738 | 0.069539 | 0.0 | 0.715403 |
| O | 0.000437 | 0.012882 | 0.014839 | 0.000324 | 0.000223 | 0.019595 | 0.021812 | 0.040195 | 0.0 | 0.889694 |

## Emission Matrix

In [5]:
```
def create_emission_matrix(sentences, tags):

    # Vocab
    vocab = set()
    for s in sentences:
        vocab.update(set(s))
    no_vocab = len(vocab)
    vocab = list(vocab)

    # Tags
    tag = set()
    for t in tags:
        tag.update(set(t))
    tag = list(tag)
    tag.remove("<s>")
    no_tag = len(tag)

    # Emission Matrix
    emission_matrix = pd.DataFrame(np.zeros((no_vocab, no_tag)), index=vocab, columns=tag)

    # Populate Transition Matrix
    no = len(sentences)
    pair = {}

    for i in range(no):
        for idx in range(len(sentences[i])):
            t = tags[i][idx]
            v = sentences[i][idx]
            if(pair.get((t, v), -1) == -1):
                pair[(t, v)] = 1
            else:
                pair[(t, v)] += 1

    for (t, v), val in pair.items():
        if(t == "<s>"):
            continue
        emission_matrix[t][v] = val

    emission_matrix = emission_matrix / emission_matrix.sum(axis=0)

    return emission_matrix

emission_matrix = create_emission_matrix(sentences, tags)
emission_matrix.T
```

Out[5]:

| | since | laloo | 290 | 1857 | duda | navigation | raised | 1,64,000 | throat | kiran | ... | re-building | undesirable | shana | higüey | antagonizes | 157 | behead | authors | thrush | vivanews.com |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| art | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| per | 0.000000 | 0.000029 | 0.000000 | 0.000000 | 0.000029 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000029 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| gpe | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| eve | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| nat | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| org | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| tim | 0.014519 | 0.000000 | 0.000000 | 0.000074 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000037 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| geo | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000022 | 0.000022 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000022 |
| O | 0.001090 | 0.000000 | 0.000003 | 0.000000 | 0.000000 | 0.000005 | 0.000106 | 0.000001 | 0.000009 | 0.000000 | ... | 0.000001 | 0.000001 | 0.000000 | 0.000000 | 0.000001 | 0.000001 | 0.000003 | 0.000003 | 0.000001 | 0.000000 |

9 rows × 31818 columns

## Viterbi Algorithm

In [6]:
```
def viterbo_algorithm(transition_matrix, emission_matrix, sent):

    start_tok = "<s>"
    tags = list(emission_matrix.columns)

    matrix = pd.DataFrame(np.zeros((len(tags), len(sent))), index=tags)
    trace_back = pd.DataFrame(np.full((len(tags), len(sent)), -1), index = tags)

    result = [0] * len(sent)

    # Initial Prob
    for t in tags:
        word = sent[0]
        matrix[0][t] = transition_matrix[start_tok][t] * emission_matrix[t][word]

    # Forward Pass
    for idx in range(1, len(sent)):
        prev_idx = idx - 1
        word = sent[idx]

        for tag in tags:
            possible_path = []
            for t in tags:
                possible_path.append(matrix[prev_idx][t] * transition_matrix[t][tag] * emission_matrix[tag][word])
            matrix[idx][tag] = max(possible_path)
            trace_back[idx][tag] = np.argmax(possible_path)

    # Backward Pass
    idx = len(sent) - 1
    last_idx = np.argmax(matrix[idx])

    while(last_idx != -1):
        result[idx] = tags[last_idx]
        last_idx = trace_back[idx][tags[last_idx]]
        idx -= 1

    return result
```

In [7]:
```
viterbo_algorithm(transition_matrix, emission_matrix, ["Indian", "womens", "team", "is", "great"])
```

Out[7]:
```
['geo', 'O', 'O', 'O', 'tim']
```