Ahmad Sharif

K436765

DATA.STAT.840 Statistical Methods for Text Data Analysis

In [32]:

```
import numpy, hmmlearn, hmmlearn.hmm
from hmmlearn import hmm
import numpy as np
```

In [31]:

```
'''Exercise 7.1: The Forward-Backward algorithm.
Consider a HMM model with the following vocabulary of 14 words: '''

vocabulary = ['a', 'the', 'over', 'beside', 'near', 'quick', 'brown', 'lazy', 'jumps', 'runs', 'walks', 'fox', 'dog', 'cat'
states = np.arange(1, 6)

concatenated_data = ['the', 'quick', 'fox', 'jumps', 'over', 'a', 'dog']

sequence = np.array([vocabulary.index(word) + 1 for word in concatenated_data]).reshape(-1, 1)


model = hmm.MultinomialHMM(n_components=10, n_iter=100, verbose=True)
model.startprob_ = np.array([1 / len(states)] * len(states))
model.transmat_ = np.array([[0.2, 0.2, 0.2, 0.2, 0.2]] * len(states))
model.emissionprob_ = np.array([[1.0] * len(vocabulary)] * len(states))


model.fit(sequence)

log_likelihood, state_sequence = model.score_samples(sequence)
probability = np.exp(log_likelihood)

print('Probability:', probability)
```

```
MultinomialHMM has undergone major changes. The previous version was implementing a CategoricalHMM (a special case of Multin
omialHMM). This new implementation follows the standard definition for a Multinomial distribution (e.g. as in https://en.wik
ipedia.org/wiki/Multinomial_distribution). See these issues for details:
https://github.com/hmmlearn/hmmlearn/issues/335
https://github.com/hmmlearn/hmmlearn/issues/340
Even though the 'startprob_' attribute is set, it will be overwritten during initialization because 'init_params' contains
's'
Even though the 'transmat_' attribute is set, it will be overwritten during initialization because 'init_params' contains
't'
Fitting a model with 99 free scalar parameters with only 7 data points will result in a degenerate solution.
Probability: 1.0
         1      -0.00000000             +nan
         2       0.00000000      +0.00000000
```

Exercise 7.4: Argue with an AI chatbot about hidden Markov models

I: Hidden Markov models and the Viterbi and Baum-Welch algorithms are still important and needed in the age of neural large language models.

Do you agree ?

AI: Yes, HMMs and their associated algorithms, such as the Viterbi and Baum-Welch algorithms, are still important and needed in the age of neural large language models. They provide interpretability, are effective for limited data and time-series analysis, allow for model training from unlabeled data, and can serve as a baseline model or for data generation. Their relevance depends on the specific task requirements and available resources.

I : Markov model is better because probability of an occurring event is dependent only on the current state of a system.

Are you still firm in your opinion

Both Markov models and neural large language models have their own strengths and are useful for different tasks. Markov models excel in capturing dependencies between events based on the current state of the system, while neural large language models can capture complex patterns and long-range context. The choice depends on the specific task and data characteristics.