NLP
ASSIGNMENT-6

2403A52060

BATCH – 03

TEXT:

The HMM is based on augmenting the Markov chain. A Markov chain is a model that tells us something about the probabilities of sequences of random variables, states, each of which can take on values from some set. These sets can be words, or tags, or symbols representing anything, like the weather. A Markov chain makes a very strong assumption that if we want to predict the future in the sequence, all that matters is the current state. The states before the current state have no impact on the future except via the current state. It's as if to predict tomorrow's weather you could examine today's weather but you weren't allowed to look at yesterday's weather.

NLTK :

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
nltk.download('punkt_tab')
nltk.download('averaged_perceptron_tagger_eng')
```

OUTPUT:

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger_eng is already up-to-
[nltk_data]       date!
True
```

```
text="The HMM is based on augmenting the Markov chain. A Markov chain is a
model that tells us something about the probabilities of sequences of
random variables, states, each of which can take on values from some set.
These sets can be words, or tags, or symbols representing anything, like
the weather. A Markov chain makes a very strong assumption that if we want
to predict the future in the sequence, all that matters is the current
state. The states before the current state have no impact on the future
except via the current state. It's as if to predict tomorrow's weather you
could examine today's weather but you weren't allowed to look at
yesterday's weather."
tokens = word_tokenize(text)
stop_words = set(stopwords.words('english'))
filtered = [w for w in tokens if w.lower() not in stop_words] # Corrected
'stops' to 'stop_words'
lemmatizer = WordNetLemmatizer()
lemmas = [lemmatizer.lemmatize(w) for w in filtered]
pos_tags = nltk.pos_tag(lemmas)
print("After tokenization: ",tokens)
print("After filtered: ",filtered)
print("After lemmatization: ",lemmas)
print("After pos_tags:  ",pos_tags)
```

OUTPUT:

After tokenization: ['The', 'HMM', 'is', 'based', 'on', 'augmenting', 'the', 'Markov', 'chain', '.', 'A', 'Markov', 'chain', 'is', 'a', 'model', 'that', 'tells', 'us', 'something', 'about', 'the', 'probabilities', 'of', 'sequences', 'of', 'random', 'variables', ',', 'states', ',', 'each', 'of', 'which', 'can', 'take', 'on', 'values', 'from', 'some', 'set', '.', 'These', 'sets', 'can', 'be', 'words', ',', 'or', 'tags', ',', 'or', 'symbols', 'representing', 'anything', ',', 'like', 'the', 'weather', '.', 'A', 'Markov', 'chain', 'makes', 'a', 'very', 'strong', 'assumption', 'that', 'if', 'we', 'want', 'to', 'predict', 'the', 'future', 'in', 'the', 'sequence', ',', 'all', 'that', 'matters', 'is', 'the', 'current', 'state', '.', 'The', 'states', 'before', 'the', 'current', 'state', 'have', 'no', 'impact', 'on', 'the', 'future', 'except', 'via', 'the', 'current', 'state', '.', 'It', '’', 's', 'as', 'if', 'to', 'predict', 'tomorrow', '’', 's', 'weather', 'you', 'could', 'examine', 'today', '’', 's', 'weather', 'but',

'you', 'weren', '’', 't', 'allowed', 'to', 'look', 'at', 'yesterday', '’', 's', 'weather', '.']

After filtered: ['HMM', 'based', 'augmenting', 'Markov', 'chain', '.', 'Markov', 'chain', 'model', 'tells', 'us', 'something', 'probabilities', 'sequences', 'random', 'variables', ',', 'states', ',', 'take', 'values', 'set', '.', 'sets', 'words', ',', 'tags', ',', 'symbols', 'representing', 'anything', ',', 'like', 'weather', '.', 'Markov', 'chain', 'makes', 'strong', 'assumption', 'want', 'predict', 'future', 'sequence', ',', 'matters', 'current', 'state', '.', 'states', 'current', 'state', 'impact', 'future', 'except', 'via', 'current', 'state', '.', '’', 'predict', 'tomorrow', '’', 'weather', 'could', 'examine', 'today', '’', 'weather', '’', 'allowed', 'look', 'yesterday', '’', 'weather', '.']

After lemmatization: ['HMM', 'based', 'augmenting', 'Markov', 'chain', '.', 'Markov', 'chain', 'model', 'tell', 'u', 'something', 'probability', 'sequence', 'random', 'variable', ',', 'state', ',', 'take', 'value', 'set', '.', 'set', 'word', ',', 'tag', ',', 'symbol', 'representing', 'anything', ',', 'like', 'weather', '.', 'Markov', 'chain', 'make', 'strong', 'assumption', 'want', 'predict', 'future', 'sequence', ',', 'matter', 'current', 'state', '.', 'state', 'current', 'state', 'impact', 'future', 'except', 'via', 'current', 'state', '.', '’', 'predict', 'tomorrow', '’', 'weather', 'could', 'examine', 'today', '’', 'weather', '’', 'allowed', 'look', 'yesterday', '’', 'weather', '.']

After pos_tags: [('HMM', 'NNP'), ('based', 'VBN'), ('augmenting', 'VBG'), ('Markov', 'NNP'), ('chain', 'NN'), ('.', '.'), ('Markov', 'NNP'), ('chain', 'NN'), ('model', 'NN'), ('tell', 'VBP'), ('u', 'JJ'), ('something', 'NN'), ('probability', 'NN'), ('sequence', 'NN'), ('random', 'NN'), ('variable', 'JJ'), (',', ','), ('state', 'NN'), (',', ','), ('take', 'VB'), ('value', 'NN'), ('set', 'VBN'), ('.', '.'), ('set', 'VB'), ('word', 'NN'), (',', ','), ('tag', 'NN'), (',', ','), ('symbol', 'NN'), ('representing', 'VBG'), ('anything', 'NN'), (',', ','), ('like', 'IN'), ('weather', 'NN'), ('.', '.'), ('Markov', 'NNP'), ('chain', 'NN'), ('make', 'VBP'), ('strong', 'JJ'), ('assumption', 'NN'), ('want', 'VBP'), ('predict', 'JJ'), ('future', 'NN'), ('sequence', 'NN'), (',', ','), ('matter', 'NN'), ('current', 'JJ'), ('state', 'NN'), ('.', '.'), ('state', 'NN'), ('current', 'JJ'), ('state', 'NN'), ('impact', 'NN'), ('future', 'JJ'), ('except', 'IN'), ('via', 'IN'), ('current', 'JJ'), ('state', 'NN'), ('.', '.'), ('’', 'CC'), ('predict', 'JJ'), ('tomorrow', 'NN'), ('’', 'NN'), ('weather', 'NN'), ('could', 'MD'), ('examine', 'VB'), ('today', 'NN'), ('’', 'NNP'), ('weather', 'NN'), ('’', 'NN'), ('allowed', 'VBN'), ('look', 'NN'), ('yesterday', 'NN'), ('’', 'IN'), ('weather', 'NN'), ('.', '.')]

SPACY:

```
import spacy
nlp = spacy.load("en_core_web_sm")
text = "The HMM is based on augmenting the Markov chain. A Markov chain is
a model that tells us something about the probabilities of sequences of
random variables, states, each of which can take on values from some set.
These sets can be words, or tags, or symbols representing anything, like
```

```
the weather. A Markov chain makes a very strong assumption that if we want
to predict the future in the sequence, all that matters is the current
state. The states before the current state have no impact on the future
except via the current state. It's as if to predict tomorrow's weather you
could examine today's weather but you weren't allowed to look at
yesterday's weather."
doc = nlp(text)
print("\n--- Named Entity Recognition (NER) ---")
for ent in doc.ents:
    print(f"Entity: {ent.text}, Label: {ent.label_}")
print("\n--- Dependency Parsing (Head & Dependency Relation) ---")
for token in doc:
    print(f"Token: {token.text}\t-> Head: {token.head.text}\t-> Dep:
{token.dep_}")
print("\n--- Filtering Tokens (No Punctuation or Digits) ---")
filtered_tokens_spacy = [token.text for token in doc if not token.is_punct
and not token.is_digit]
print("Filtered tokens (spaCy): ", filtered_tokens_spacy)
print("\n--- Keeping Only Alphabetic Tokens ---")
alphabetic_tokens_spacy = [token.text for token in doc if token.is_alpha]
print("Alphabetic tokens (spaCy): ", alphabetic_tokens_spacy)
```

OUTPUT:

--- Named Entity Recognition (NER) ---

```
Entity: HMM, Label: ORG
Entity: Markov, Label: PERSON
Entity: Markov, Label: PERSON
Entity: Markov, Label: PERSON
Entity: tomorrow, Label: DATE
Entity: today, Label: DATE
Entity: yesterday, Label: DATE
```

--- Dependency Parsing (Head & Dependency Relation) ---

```
Token: The      -> Head: HMM    -> Dep: det
Token: HMM      -> Head: based  -> Dep: nsubjpass
Token: is       -> Head: based  -> Dep: auxpass
Token: based    -> Head: based  -> Dep: ROOT
Token: on       -> Head: based  -> Dep: prep
Token: augmenting       -> Head: on     -> Dep: pcomp
Token: the      -> Head: chain  -> Dep: det
Token: Markov   -> Head: chain  -> Dep: compound
Token: chain    -> Head: augmenting     -> Dep: dobj
Token: .        -> Head: based  -> Dep: punct
Token: A        -> Head: chain  -> Dep: det
Token: Markov   -> Head: chain  -> Dep: compound
```

```
Token: chain    -> Head: is     -> Dep: nsubj
Token: is       -> Head: is     -> Dep: ROOT
Token: a        -> Head: model  -> Dep: det
Token: model    -> Head: is     -> Dep: attr
Token: that     -> Head: tells  -> Dep: nsubj
Token: tells    -> Head: model  -> Dep: relcl
Token: us       -> Head: tells  -> Dep: dative
Token: something        -> Head: tells  -> Dep: dobj
Token: about    -> Head: something      -> Dep: prep
Token: the      -> Head: probabilities -> Dep: det
Token: probabilities    -> Head: about  -> Dep: pobj
Token: of       -> Head: probabilities -> Dep: prep
Token: sequences        -> Head: of     -> Dep: pobj
Token: of       -> Head: sequences      -> Dep: prep
Token: random   -> Head: variables      -> Dep: amod
Token: variables        -> Head: of     -> Dep: pobj
Token: ,        -> Head: sequences      -> Dep: punct
Token: states   -> Head: sequences      -> Dep: conj
Token: ,        -> Head: states -> Dep: punct
Token: each     -> Head: take   -> Dep: nsubj
Token: of       -> Head: each   -> Dep: prep
Token: which    -> Head: of     -> Dep: pobj
Token: can      -> Head: take   -> Dep: aux
Token: take     -> Head: probabilities -> Dep: relcl
Token: on       -> Head: take   -> Dep: prt
Token: values   -> Head: take   -> Dep: dobj
Token: from     -> Head: take   -> Dep: prep
Token: some     -> Head: set    -> Dep: det
Token: set      -> Head: from   -> Dep: pobj
Token: .        -> Head: is     -> Dep: punct
Token: These    -> Head: sets   -> Dep: det
Token: sets     -> Head: be     -> Dep: nsubj
Token: can      -> Head: be     -> Dep: aux
Token: be       -> Head: be     -> Dep: ROOT
Token: words    -> Head: be     -> Dep: attr
Token: ,        -> Head: words  -> Dep: punct
Token: or       -> Head: words  -> Dep: cc
Token: tags     -> Head: words  -> Dep: conj
Token: ,        -> Head: tags   -> Dep: punct
Token: or       -> Head: tags   -> Dep: cc
Token: symbols  -> Head: tags   -> Dep: conj
Token: representing     -> Head: symbols        -> Dep: acl
Token: anything -> Head: representing   -> Dep: dobj
Token: ,        -> Head: words  -> Dep: punct
Token: like     -> Head: words  -> Dep: prep
Token: the      -> Head: weather        -> Dep: det
Token: weather  -> Head: like   -> Dep: pobj
Token: .        -> Head: be     -> Dep: punct
Token: A        -> Head: chain  -> Dep: det
Token: Markov   -> Head: chain  -> Dep: compound
Token: chain    -> Head: makes  -> Dep: nsubj
Token: makes    -> Head: makes  -> Dep: ROOT
Token: a        -> Head: assumption     -> Dep: det
Token: very     -> Head: strong -> Dep: advmod
Token: strong   -> Head: assumption     -> Dep: amod
Token: assumption       -> Head: makes  -> Dep: dobj
Token: that     -> Head: is     -> Dep: mark
```

```
Token: if       -> Head: want   -> Dep: mark
Token: we       -> Head: want   -> Dep: nsubj
Token: want     -> Head: is     -> Dep: advcl
Token: to       -> Head: predict        -> Dep: aux
Token: predict  -> Head: want   -> Dep: xcomp
Token: the      -> Head: future -> Dep: det
Token: future   -> Head: predict        -> Dep: dobj
Token: in       -> Head: predict        -> Dep: prep
Token: the      -> Head: sequence       -> Dep: det
Token: sequence -> Head: in     -> Dep: pobj
Token: ,        -> Head: is     -> Dep: punct
Token: all      -> Head: is     -> Dep: nsubj
Token: that     -> Head: matters        -> Dep: nsubj
Token: matters  -> Head: all    -> Dep: relcl
Token: is       -> Head: assumption     -> Dep: acl
Token: the      -> Head: state  -> Dep: det
Token: current  -> Head: state  -> Dep: amod
Token: state    -> Head: is     -> Dep: attr
Token: .        -> Head: makes  -> Dep: punct
Token: The      -> Head: states -> Dep: det
Token: states   -> Head: have   -> Dep: nsubj
Token: before   -> Head: have   -> Dep: mark
Token: the      -> Head: state  -> Dep: det
Token: current  -> Head: state  -> Dep: amod
Token: state    -> Head: before -> Dep: pobj
Token: have     -> Head: have   -> Dep: ROOT
Token: no       -> Head: impact -> Dep: det
Token: impact   -> Head: have   -> Dep: dobj
Token: on       -> Head: impact -> Dep: prep
Token: the      -> Head: future -> Dep: det
Token: future   -> Head: on     -> Dep: pobj
Token: except   -> Head: have   -> Dep: prep
Token: via      -> Head: except -> Dep: prep
Token: the      -> Head: state  -> Dep: det
Token: current  -> Head: state  -> Dep: amod
Token: state    -> Head: via    -> Dep: pobj
Token: .        -> Head: have   -> Dep: punct
Token: It       -> Head: ’s     -> Dep: nsubj
Token: ’s       -> Head: ’s     -> Dep: ROOT
Token: as       -> Head: predict        -> Dep: mark
Token: if       -> Head: predict        -> Dep: mark
Token: to       -> Head: predict        -> Dep: aux
Token: predict  -> Head: examine        -> Dep: advcl
Token: tomorrow -> Head: weather        -> Dep: poss
Token: ’s       -> Head: tomorrow       -> Dep: case
Token: weather  -> Head: predict        -> Dep: dobj
Token: you      -> Head: examine        -> Dep: nsubj
Token: could    -> Head: examine        -> Dep: aux
Token: examine  -> Head: ’s     -> Dep: ccomp
Token: today    -> Head: weather        -> Dep: poss
Token: ’s       -> Head: today  -> Dep: case
Token: weather  -> Head: examine        -> Dep: dobj
Token: but      -> Head: ’s     -> Dep: cc
Token: you      -> Head: allowed        -> Dep: nsubjpass
Token: were     -> Head: allowed        -> Dep: auxpass
Token: n’t      -> Head: allowed        -> Dep: neg
Token: allowed  -> Head: ’s     -> Dep: conj
```

```
Token: to       -> Head: look   -> Dep: aux
Token: look     -> Head: allowed        -> Dep: xcomp
Token: at       -> Head: look   -> Dep: prep
Token: yesterday        -> Head: weather        -> Dep: poss
Token: ’s       -> Head: yesterday      -> Dep: case
Token: weather  -> Head: at     -> Dep: pobj
Token: .        -> Head: allowed        -> Dep: punct
```

--- Filtering Tokens (No Punctuation or Digits) ---

```
Filtered tokens (spaCy):  ['The', 'HMM', 'is', 'based', 'on', 'augmenting',
'the', 'Markov', 'chain', 'A', 'Markov', 'chain', 'is', 'a', 'model', 'that',
'tells', 'us', 'something', 'about', 'the', 'probabilities', 'of',
'sequences', 'of', 'random', 'variables', 'states', 'each', 'of', 'which',
'can', 'take', 'on', 'values', 'from', 'some', 'set', 'These', 'sets', 'can',
'be', 'words', 'or', 'tags', 'or', 'symbols', 'representing', 'anything',
'like', 'the', 'weather', 'A', 'Markov', 'chain', 'makes', 'a', 'very',
'strong', 'assumption', 'that', 'if', 'we', 'want', 'to', 'predict', 'the',
'future', 'in', 'the', 'sequence', 'all', 'that', 'matters', 'is', 'the',
'current', 'state', 'The', 'states', 'before', 'the', 'current', 'state',
'have', 'no', 'impact', 'on', 'the', 'future', 'except', 'via', 'the',
'current', 'state', 'It', '’s', 'as', 'if', 'to', 'predict', 'tomorrow',
'’s', 'weather', 'you', 'could', 'examine', 'today', '’s', 'weather', 'but',
'you', 'were', 'n’t', 'allowed', 'to', 'look', 'at', 'yesterday', '’s',
'weather']
```

--- Keeping Only Alphabetic Tokens ---

```
Alphabetic tokens (spaCy):  ['The', 'HMM', 'is', 'based', 'on', 'augmenting',
'the', 'Markov', 'chain', 'A', 'Markov', 'chain', 'is', 'a', 'model', 'that',
'tells', 'us', 'something', 'about', 'the', 'probabilities', 'of',
'sequences', 'of', 'random', 'variables', 'states', 'each', 'of', 'which',
'can', 'take', 'on', 'values', 'from', 'some', 'set', 'These', 'sets', 'can',
'be', 'words', 'or', 'tags', 'or', 'symbols', 'representing', 'anything',
'like', 'the', 'weather', 'A', 'Markov', 'chain', 'makes', 'a', 'very',
'strong', 'assumption', 'that', 'if', 'we', 'want', 'to', 'predict', 'the',
'future', 'in', 'the', 'sequence', 'all', 'that', 'matters', 'is', 'the',
'current', 'state', 'The', 'states', 'before', 'the', 'current', 'state',
'have', 'no', 'impact', 'on', 'the', 'future', 'except', 'via', 'the',
'current', 'state', 'It', 'as', 'if', 'to', 'predict', 'tomorrow', 'weather',
'you', 'could', 'examine', 'today', 'weather', 'but', 'you', 'were',
'allowed', 'to', 'look', 'at', 'yesterday', 'weather']
```

HMM:

```
from nltk.tag import hmm
from nltk.corpus import treebank
import nltk
nltk.download('treebank') # Download the treebank corpus
train_data = treebank.tagged_sents()[:3000]
test_data = treebank.tagged_sents()[3000:]
trainer = hmm.HiddenMarkovModelTrainer()
```

```
hmm_tagger = trainer.train(train_data)
print(hmm_tagger.tag("Sequence modeling is important".split()))
```

OUTPUT:

```
[nltk_data] Downloading package treebank to /root/nltk_data...
[nltk_data]   Unzipping corpora/treebank.zip.
/usr/local/lib/python3.12/dist-packages/nltk/tag/hmm.py:333: RuntimeWarning:
overflow encountered in cast
  X[i, j] = self._transitions[si].logprob(self._states[j])
/usr/local/lib/python3.12/dist-packages/nltk/tag/hmm.py:335: RuntimeWarning:
overflow encountered in cast
  O[i, k] = self._output_logprob(si, self._symbols[k])
/usr/local/lib/python3.12/dist-packages/nltk/tag/hmm.py:331: RuntimeWarning:
overflow encountered in cast
  P[i] = self._priors.logprob(si)
[('Sequence', 'NNP'), ('modeling', 'NNP'), ('is', 'NNP'), ('important',
'NNP')]
/usr/local/lib/python3.12/dist-packages/nltk/tag/hmm.py:363: RuntimeWarning:
overflow encountered in cast
  O[i, k] = self._output_logprob(si, self._symbols[k])
```

```
hmm_pos_tags = hmm_tagger.tag(lemmas)
print("HMM POS Tags for the lemmatized text:")
print(hmm_pos_tags)
```

OUTPUT:

HMM POS Tags for the lemmatized text:

```
[('HMM', 'NNP'), ('based', 'NNP'), ('augmenting', 'NNP'), ('Markov', 'NNP'),
('chain', 'NNP'), ('.', 'NNP'), ('Markov', 'NNP'), ('chain', 'NNP'),
('model', 'NNP'), ('tell', 'NNP'), ('u', 'NNP'), ('something', 'NNP'),
('probability', 'NNP'), ('sequence', 'NNP'), ('random', 'NNP'), ('variable',
'NNP'), (',', 'NNP'), ('state', 'NNP'), (',', 'NNP'), ('take', 'NNP'),
('value', 'NNP'), ('set', 'NNP'), ('.', 'NNP'), ('set', 'NNP'), ('word',
'NNP'), (',', 'NNP'), ('tag', 'NNP'), (',', 'NNP'), ('symbol', 'NNP'),
('representing', 'NNP'), ('anything', 'NNP'), (',', 'NNP'), ('like', 'NNP'),
('weather', 'NNP'), ('.', 'NNP'), ('Markov', 'NNP'), ('chain', 'NNP'),
('make', 'NNP'), ('strong', 'NNP'), ('assumption', 'NNP'), ('want', 'NNP'),
('predict', 'NNP'), ('future', 'NNP'), ('sequence', 'NNP'), (',', 'NNP'),
('matter', 'NNP'), ('current', 'NNP'), ('state', 'NNP'), ('.', 'NNP'),
('state', 'NNP'), ('current', 'NNP'), ('state', 'NNP'), ('impact', 'NNP'),
('future', 'NNP'), ('except', 'NNP'), ('via', 'NNP'), ('current', 'NNP'),
('state', 'NNP'), ('.', 'NNP'), ('’', 'NNP'), ('predict', 'NNP'),
('tomorrow', 'NNP'), ('’', 'NNP'), ('weather', 'NNP'), ('could', 'NNP'),
('examine', 'NNP'), ('today', 'NNP'), ('’', 'NNP'), ('weather', 'NNP'), ('’',
'NNP'), ('allowed', 'NNP'), ('look', 'NNP'), ('yesterday', 'NNP'), ('’',
'NNP'), ('weather', 'NNP'), ('.', 'NNP')]
/usr/local/lib/python3.12/dist-packages/nltk/tag/hmm.py:363: RuntimeWarning:
overflow encountered in cast
```

```
    O[i, k] = self._output_logprob(si, self._symbols[k])
```

```
sentence = "The HMM is based on augmenting the Markov chain. A Markov
chain is a model that tells us something about the probabilities of
sequences of random variables, states, each of which can take on values
from some set. These sets can be words, or tags, or symbols representing
anything, like the weather. A Markov chain makes a very strong assumption
that if we want to predict the future in the sequence, all that matters is
the current state. The states before the current state have no impact on
the future except via the current state. It’s as if to predict tomorrow’s
weather you could examine today’s weather but you weren’t allowed to look
at yesterday’s weather."
tokens = word_tokenize(sentence)
print("Original Sentence:", sentence)
print("Tokenized Sentence:", tokens)
```

OUTPUT:

Original Sentence: The HMM is based on augmenting the Markov chain. A Markov chain is a model that tells us something about the probabilities of sequences of random variables, states, each of which can take on values from some set. These sets can be words, or tags, or symbols representing anything, like the weather. A Markov chain makes a very strong assumption that if we want to predict the future in the sequence, all that matters is the current state. The states before the current state have no impact on the future except via the current state. It’s as if to predict tomorrow’s weather you could examine today’s weather but you weren’t allowed to look at yesterday’s weather.

Tokenized Sentence: ['The', 'HMM', 'is', 'based', 'on', 'augmenting', 'the', 'Markov', 'chain', '.', 'A', 'Markov', 'chain', 'is', 'a', 'model', 'that', 'tells', 'us', 'something', 'about', 'the', 'probabilities', 'of', 'sequences', 'of', 'random', 'variables', ',', 'states', ',', 'each', 'of', 'which', 'can', 'take', 'on', 'values', 'from', 'some', 'set', '.', 'These', 'sets', 'can', 'be', 'words', ',', 'or', 'tags', ',', 'or', 'symbols', 'representing', 'anything', ',', 'like', 'the', 'weather', '.', 'A', 'Markov', 'chain', 'makes', 'a', 'very', 'strong', 'assumption', 'that', 'if', 'we', 'want', 'to', 'predict', 'the', 'future', 'in', 'the', 'sequence', ',', 'all', 'that', 'matters', 'is', 'the', 'current', 'state', '.', 'The', 'states', 'before', 'the', 'current', 'state', 'have', 'no', 'impact', 'on', 'the', 'future', 'except', 'via', 'the', 'current', 'state', '.', 'It', '’', 's', 'as', 'if', 'to', 'predict', 'tomorrow', '’', 's', 'weather', 'you', 'could', 'examine', 'today', '’', 's', 'weather', 'but', 'you', 'weren', '’', 't', 'allowed', 'to', 'look', 'at', 'yesterday', '’', 's', 'weather', '.']