

Title: Sequence Modeling with HMMs for Technical Text.

Dataset Source:

- o POS-tagged sample abstracts
- o (Abstracts sourced from arXiv via Kaggle)

Lab Objectives:

- o Understand domain mismatch in HMMs.
- o Analyze tag transition patterns in technical writing

Tasks:

- o Collect 20–30 research abstracts.
- o Automatically POS-tag them using NLTK.
- o Treat the tagged data as training data for HMM.

Compute:

- o Transition probabilities
- o Emission probabilities
- o Analyze:
 - o Most frequent tag transitions
 - o Apply HMM tagging to a new abstract sentence.

Expected Output:

- o Transition matrix
- o Emission probability examples.
- o Analysis of domain-specific POS patterns

STEPS:

1.Convert zip file into unzip-csv file

```
import zipfile

zip_path = "/content/arxiv_data.csv.zip"
extract_path = "/content/arxiv_extracted"

os.makedirs(extract_path, exist_ok=True)

with zipfile.ZipFile(zip_path) as z:
    print("Files inside zip:", z.namelist())
    z.extractall(extract_path)
```

```
Files inside zip: ['arxiv_data.csv']
```

```
import zipfile
import os

zip_path = "/content/arxiv_data.csv.zip"
extract_path = "/content/arxiv_extracted"

os.makedirs(extract_path, exist_ok=True)

with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)

os.listdir(extract_path)
```

```
['arxiv_data.csv']
```

2.Install & Import Required Libraries

```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk) (8.3.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (from nltk) (1.5.3)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2025.11.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.1)
```

```
import pandas as pd
import nltk
from nltk import word_tokenize, pos_tag
from nltk.tag.hmm import HiddenMarkovModelTrainer
from collections import defaultdict, Counter
```

```
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('punkt_tab')
nltk.download('averaged_perceptron_tagger_eng')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt 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
```

3.Load arXiv CSV File

```
import pandas as pd

csv_path = "/content/arxiv_extracted/arxiv_data.csv"

df = pd.read_csv(csv_path, encoding="latin1")
df.head()
```

	titles	summaries	terms
0	Survey on Semantic Stereo Matching / Semantic ...	Stereo matching is one of the widely used tech...	['cs.CV', 'cs.LG']
1	FUTURE-AI: Guiding Principles and Consensus Re...	The recent advancements in artificial intellig...	['cs.CV', 'cs.AI', 'cs.LG']
2	Enforcing Mutual Consistency of Hard Regions f...	In this paper, we proposed a novel mutual cons...	['cs.CV', 'cs.AI']
3	Parameter Decoupling Strategy for Semi-supervi...	Consistency training has proven to be an advan...	['cs.CV']
4	Background-Foreground Segmentation for Interio...	To ensure safety in automated driving, the cor...	['cs.CV', 'cs.LG']

4.Collect 20–30 Research Abstracts

```
abstracts = df.iloc[:25, 0].dropna().astype(str).tolist()
len(abstracts)
```

25

5.POS Tagging Using NLTK

```
tagged_sentences = []

for abstract in abstracts:
    sentences = nltk.sent_tokenize(abstract)
    for sent in sentences:
        tokens = nltk.word_tokenize(sent)
        pos_tags = nltk.pos_tag(tokens)
        tagged_sentences.append(pos_tags)

tagged_sentences[:2]
```

```
[(['Survey', 'NNP'),
 ('on', 'IN'),
```

```
(('Semantic', 'NNP'),
 ('Stereo', 'NNP'),
 ('Matching', 'NNP'),
 ('/', 'NNP'),
 ('Semantic', 'NNP'),
 ('Depth', 'NNP'),
 ('Estimation', 'NNP']],
 [('FUTURE-AI', 'NN'),
 (':', ':'),
 ('Guiding', 'NNP'),
 ('Principles', 'NNP'),
 ('and', 'CC'),
 ('Consensus', 'NNP'),
 ('Recommendations', 'NNP'),
 ('for', 'IN'),
 ('Trustworthy', 'NNP'),
 ('Artificial', 'NNP'),
 ('Intelligence', 'NNP'),
 ('in', 'IN'),
 ('Future', 'NNP'),
 ('Medical', 'NNP'),
 ('Imaging', 'NNP')]]
```

6.Prepare Data for HMM Training

```
tag_sequences = []
word_tag_pairs = []

for sentence in tagged_sentences:
    tags = []
    for word, tag in sentence:
        tags.append(tag)
        word_tag_pairs.append((tag, word.lower()))
    tag_sequences.append(tags)
```

7.Compute Transition Probabilitie.

```
transition_counts = defaultdict(Counter)

for tags in tag_sequences:
    for i in range(len(tags) - 1):
        transition_counts[tags[i]][tags[i+1]] += 1
```

```
transition_probs = {}

for tag, next_tags in transition_counts.items():
    total = sum(next_tags.values())
    transition_probs[tag] = {
        next_tag: count / total
        for next_tag, count in next_tags.items()
    }
```

8.Transition Matrix

```
all_tags = sorted(set(tag for seq in tag_sequences for tag in seq))

transition_matrix = pd.DataFrame(0.0, index=all_tags, columns=all_tags)

for tag, transitions in transition_probs.items():
    for next_tag, prob in transitions.items():
        transition_matrix.loc[tag, next_tag] = prob

transition_matrix.head()
```

	,	:	CC	CD	DT	IN	JJ	NN	NNP	NNPS	NNS	RB	TO	VBD	VBG
,	0.0	0.0	0.333333	0.0	0.000000	0.0	0.000000	0.0	0.666667	0.0	0.0	0.0	0.0	0.0	0.000000
:	0.0	0.0	0.000000	0.0	0.272727	0.0	0.272727	0.0	0.272727	0.0	0.0	0.0	0.0	0.0	0.181818
CC	0.0	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.0	1.000000	0.0	0.0	0.0	0.0	0.0	0.000000
CD	0.0	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.0	1.000000	0.0	0.0	0.0	0.0	0.0	0.000000
DT	0.0	0.0	0.000000	0.0	0.000000	0.0	0.500000	0.0	0.500000	0.0	0.0	0.0	0.0	0.0	0.000000

9. Compute Emission Probabilities

```
emission_counts = defaultdict(Counter)

for tag, word in word_tag_pairs:
    emission_counts[tag][word] += 1
```

```
emission_probs = {}

for tag, words in emission_counts.items():
    total = sum(words.values())
    emission_probs[tag] = {
        word: count / total
        for word, count in words.items()
    }
```

```
Counter(emission_counts['NN']).most_common(10)
```

```
[('future-ai', 1),
 ('industry', 1),
 ('edgeflow', 1),
 ('meta-learning', 1),
 ('missformer', 1),
 ('context', 1),
 ('sensing', 1),
 ('case-study', 1),
 ('image', 1),
 ('segmentation', 1)]
```

10. Most Frequent POS Tag Transitions

```
transition_frequency = []

for tag, next_tags in transition_counts.items():
    for next_tag, count in next_tags.items():
        transition_frequency.append((tag, next_tag, count))

transition_frequency = sorted(transition_frequency, key=lambda x: x[2], reverse=True)

transition_frequency[:10]
```

```
[('NNP', 'NNP', 77),
 ('NNP', 'IN', 26),
 ('IN', 'NNP', 26),
 ('JJ', 'NNP', 12),
 ('NN', ':', 8),
 ('JJ', 'NN', 6),
 ('CC', 'NNP', 5),
 ('NNP', 'CC', 4),
 ('DT', 'NNP', 4),
 ('DT', 'JJ', 4)]
```

11. Apply HMM Tagging to a New Abstract Sentence

```
test_sentence = "Hidden Markov models are widely used in natural language processing"

tokens = nltk.word_tokenize(test_sentence)
predicted_tags = nltk.pos_tag(tokens)

list(zip(tokens, predicted_tags))
```

```
[('Hidden', ('Hidden', 'NNP')),
 ('Markov', ('Markov', 'NNP')),
 ('models', ('models', 'NNS')),
 ('are', ('are', 'VBP')),
 ('widely', ('widely', 'RB')),
 ('used', ('used', 'VBN')),
 ('in', ('in', 'IN')),
 ('natural', ('natural', 'JJ')),
 ('language', ('language', 'NN')),
 ('processing', ('processing', 'NN'))]
```

12. Domain-Specific POS Pattern Analysis

Analysis: Technical research abstracts show frequent NN → NN transitions due to compound nouns such as neural network model.

Adjectives (JJ) commonly precede nouns (NN) to describe methods and architectures.

Verbs are less frequent compared to nouns, reflecting the information-dense style of scientific writing.