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

```
import pandas as pd
import random

# Load dataset
df = pd.read_csv("arxiv_data_210930-054931.csv", engine='python', on_bad_lines='warn')

# Drop missing abstracts
df = df.dropna(subset=["abstracts"])
```

```
# Randomly sample 25 abstracts
abstracts = df["abstracts"].sample(n=25, random_state=42).tolist()
```

```
/tmp/ipython-input-3834592860.py:5: ParserWarning: Skipping line 2436: unexpected end of data

  df = pd.read_csv("arxiv_data_210930-054931.csv", engine='python', on_bad_lines='warn')
```

```
abstracts = df[
    df["abstracts"].str.len() > 500
]["abstracts"].sample(n=25, random_state=42).tolist()
```

```
import nltk
from nltk import sent_tokenize, word_tokenize, pos_tag

nltk.download("punkt")
nltk.download("averaged_perceptron_tagger_eng") # Updated to download the specific English tagger
nltk.download("punkt_tab")

tagged_sentences = []

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

```
[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_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
```

```
from nltk.tag import hmm

trainer = hmm.HiddenMarkovModelTrainer()
hmm_tagger = trainer.train(tagged_sentences)
```

```
transitions = hmm_tagger._transitions

for prev_tag in transitions:
    for next_tag in transitions[prev_tag].samples(): # Iterate over samples (next_tags)
        prob = transitions[prev_tag].prob(next_tag) # Get probability for each sample
```

```
        if prob > 0.05:
            print(f"{prev_tag} → {next_tag}: {prob:.3f}")
```

Show hidden output

```
emissions = hmm_tagger._outputs

def top_emissions(tag, n=5):
    tag_emissions = []
    if tag in emissions:
        for word in emissions[tag].samples():
            prob = emissions[tag].prob(word)
            tag_emissions.append((word, prob))
    return sorted(
        tag_emissions,
        key=lambda x: x[1],
        reverse=True
    )[:n]

print("NN:", top_emissions("NN"))
print("JJ:", top_emissions("JJ"))
```

```
odel', 0.030413625304136254), ('language', 0.0267639902676399), ('learning', 0.024330900243309004), ('image', 0.021897810218978103), ('approach', 0.019464
atural', 0.03155818540433925), ('different', 0.02564102564102564), ('such', 0.01972386587771203), ('state-of-the-art', 0.01972386587771203), ('new', 0.017
```

```
print(type(tagged_sentences[0]))
print(tagged_sentences[0])
print(type(tagged_sentences[0][0]))
```

```
<class 'list'>
[('Can', 'MD'), ('we', 'PRP'), ('teach', 'VB'), ('a', 'DT'), ('robot', 'NN'), ('to', 'TO'), ('recognize', 'VB'), ('and', 'CC'), ('make', 'VB'), ('predict
<class 'tuple'>
```

```
clean_tagged_sentences = []

for sent in tagged_sentences:
    clean_sent = []
    for item in sent:
        if (
            isinstance(item, tuple) and
            len(item) == 2 and
            isinstance(item[0], str) and
            isinstance(item[1], str)
```

```
        ):
            clean_sent.append(item)
    if len(clean_sent) > 0:
        clean_tagged_sentences.append(clean_sent)
```

```
from nltk.tag import hmm

trainer = hmm.HiddenMarkovModelTrainer()
hmm_tagger = trainer.train(clean_tagged_sentences)
```

```
trainer = hmm.HiddenMarkovModelTrainer()
hmm_tagger = trainer.train(
    clean_tagged_sentences,
    estimator=lambda fd, bins: nltk.LaplaceProbDist(fd, bins)
)
```

```
from nltk import word_tokenize

test_sentence = "We propose an efficient neural network architecture"
tokens = word_tokenize(test_sentence)

# Ensure tokens are strings (paranoia check)
tokens = [str(t) for t in tokens]

print(hmm_tagger.tag(tokens))
```

```
[('We', 'PRP'), ('propose', 'VBP'), ('an', 'DT'), ('efficient', 'JJ'), ('neural', 'JJ'), ('network', 'NN'), ('architecture', 'NN')]
```

```
import pandas as pd

cs_df = df[df["terms"].str.contains("cs")]
math_df = df[df["terms"].str.contains("math")]

# Construct the transition matrix data as a dictionary of dictionaries
transition_matrix_data = {}
for prev_tag in transitions:
    transition_matrix_data[prev_tag] = {}
    for next_tag in transitions[prev_tag].samples():
        prob = transitions[prev_tag].prob(next_tag)
        transition_matrix_data[prev_tag][next_tag] = prob
```

```
trans_df = pd.DataFrame(transition_matrix_data).fillna(0)
trans_df.to_csv("transition_matrix.csv")
print(trans_df.round(3))
```

```
VBD   0.000  0.000  0.000  0.003  0.005  0.000  0.000  0.000  0.000  0.000
CD    0.000  0.000  0.000  0.008  0.000  0.022  0.021  0.000  0.030  0.045
RBS   0.000  0.000  0.000  0.003  0.000  0.000  0.000  0.000  0.000  0.000
WDT   0.000  0.000  0.000  0.000  0.016  0.000  0.000  0.033  0.016  0.000
WRB   0.000  0.000  0.000  0.000  0.005  0.000  0.014  0.003  0.000  0.000
)     0.000  0.000  0.000  0.000  0.012  0.000  0.000  0.003  0.000  0.000
:     0.000  0.000  0.000  0.000  0.016  0.000  0.000  0.020  0.000  0.000
POS   0.000  0.000  0.000  0.000  0.001  0.000  0.000  0.003  0.000  0.000
JJR   0.000  0.000  0.000  0.000  0.000  0.011  0.000  0.000  0.002  0.000
WP$   0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.003  0.000  0.000
PDT   0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.004  0.000
RP    0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
''    0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
EX    0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
``    0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000

      ...  PDT     WP     EX    JJR  WP$    POS   ``   ''  RBS    .
PRP   ...  0.0  0.333  0.000  0.000  0.0  0.000  0.0  0.0  0.0  0.0
VB    ...  0.0  0.000  0.000  0.167  0.0  0.000  0.0  0.0  0.0  0.0
RB    ...  0.0  0.000  0.000  0.000  0.0  0.000  0.0  0.0  0.0  0.0
,     ...  0.0  0.000  0.000  0.000  0.0  0.000  0.0  0.0  0.0  0.0
```

```
)     ...  0.0  0.000  0.000  0.000  0.0  0.000  0.0  1.0  0.0  0.0
:     ...  0.0  0.000  0.000  0.000  0.0  0.000  0.0  0.0  0.0  0.0
POS   ...  0.0  0.000  0.000  0.000  0.0  0.000  0.0  0.0  0.0  0.0
JJR   ...  0.0  0.000  0.000  0.000  0.0  0.000  0.0  0.0  0.0  0.0
WP$   ...  0.0  0.000  0.000  0.000  0.0  0.000  0.0  0.0  0.0  0.0
PDT   ...  0.0  0.000  0.000  0.000  0.0  0.000  0.0  0.0  0.0  0.0
RP    ...  0.0  0.000  0.000  0.000  0.0  0.000  0.0  0.0  0.0  0.0
''    ...  0.0  0.000  0.000  0.000  0.0  0.000  0.0  0.0  0.0  0.0
EX    ...  0.0  0.000  0.000  0.000  0.0  0.000  0.0  0.0  0.0  0.0
``    ...  0.0  0.000  0.000  0.000  0.0  0.000  0.0  0.0  0.0  0.0

[39 rows x 39 columns]
```

```
print("High-Probability POS Transitions (p > 0.05):\n")

for prev_tag in trans_df.index:
    for next_tag in trans_df.columns:
        prob = trans_df.loc[prev_tag, next_tag]
        if prob > 0.05:
            print(f"{prev_tag:>4} → {next_tag:<4} : {prob:.3f}")
```

Show hidden output

```
N = 5
print(f"Top {N} transitions per POS tag:\n")

for prev_tag in trans_df.index:
    top = (
        trans_df.loc[prev_tag]
        .sort_values(ascending=False)
        .head(N)
    )
    print(f"{prev_tag}:")
    for next_tag, prob in top.items():
        if prob > 0:
            print(f"   → {next_tag}: {prob:.3f}")
```

```
Top 5 transitions per POS tag:

PRP:
   → WP: 0.333
   → WRB: 0.167
   → ,: 0.146
   → WDT: 0.049
   → :: 0.040
VB:
```

```
      → MD: 0.741
      → TO: 0.620
      → JJR: 0.167
      → RB: 0.096
      → CC: 0.077
   RB:
      → MD: 0.148
      → RBR: 0.143
      → JJS: 0.125
      → VBZ: 0.108
      → VBP: 0.096
   ,:
      → ): 0.204
      → RB: 0.191
      → NNS: 0.129
      → NN: 0.108
      → VBD: 0.077
   TO:
      → VBD: 0.154
      → RBR: 0.143
      → VBN: 0.084
      → ): 0.061
      → :: 0.040
   VBZ:
      → EX: 0.667
      → WDT: 0.366
      → RBR: 0.143
      → ): 0.082
      → RB: 0.074
   VBP:
      → PRP: 0.773
      → EX: 0.333
      → WDT: 0.244
      → NNS: 0.106
      → RB: 0.051
   MD:
      → WDT: 0.073
      → PRP: 0.040
      → CD: 0.020
      → CC: 0.014
      → NNS: 0.013
   .:
      → RP: 0.500
      → POS: 0.333
      → NNS: 0.167
      → ): 0.143
      → RBR: 0.143
   IN:
      → RB: 0.500
```