



NUS | Computing
National University
of Singapore

IT5005 Artificial Intelligence

Sirigina Rajendra Prasad
AY2025/2026: Semester 1

Language Modelling using Markov Chains

Markov Chains in NLP

- **Unigram Model:**

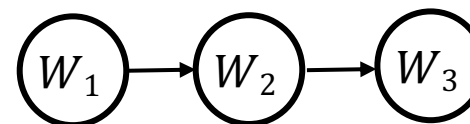
- Each word is independent



- Ex: $P(\text{"Markov Chains Rocks"}) = P(\text{"Markov"})P(\text{"Chains"})P(\text{"Rocks"})$

- **Bigram Model (First-order MC)**

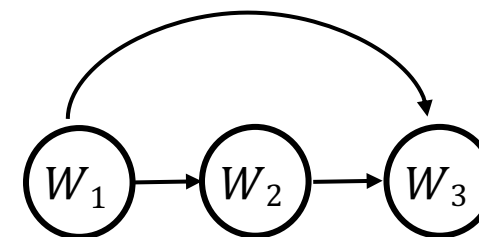
- Current word depends only on previous word



- Ex: $P(\text{"Markov Chains Rocks"}) = P(\text{"Markov"})P(\text{"Chains"}|\text{"Markov"})P(\text{"Rocks"}|\text{"Chains"})$

- **Trigram Model (Second-order MC)**

- Current word depends on previous two words



- Ex: $P(\text{"Markov Chains Rocks"}) = P(\text{"Markov"})P(\text{"Chains"}|\text{"Markov"})P(\text{"Rocks"}|\text{"Chains", "Markov"})$

Markov Chains

- Character-level Language Model
- Word-level Language Model

State Space for LMs

- Let V be the vocabulary set of tokens
- State space for n -th order Markov chains: $S = V^n$
 - V^n : all possible sequences of length n
- Number of possible states: $|V|^n$

Character-level LMs

- Assume $V = \{a, b, c, d, e\}$
 - Size of Vocabulary: $|V| = 5$
- First-order Markov Chains:
 - $n = 1$
 - Number of states: $|V|^1 = 5$
- States:
 - $S = V = \{a, b, c, d, e\}$

		Next Token				
		a	b	c	d	e
Current State	a	$P(a a)$				
	b				$P(d b)$	
	c					
	d					$P(e d)$
	e					

Character-level LMs

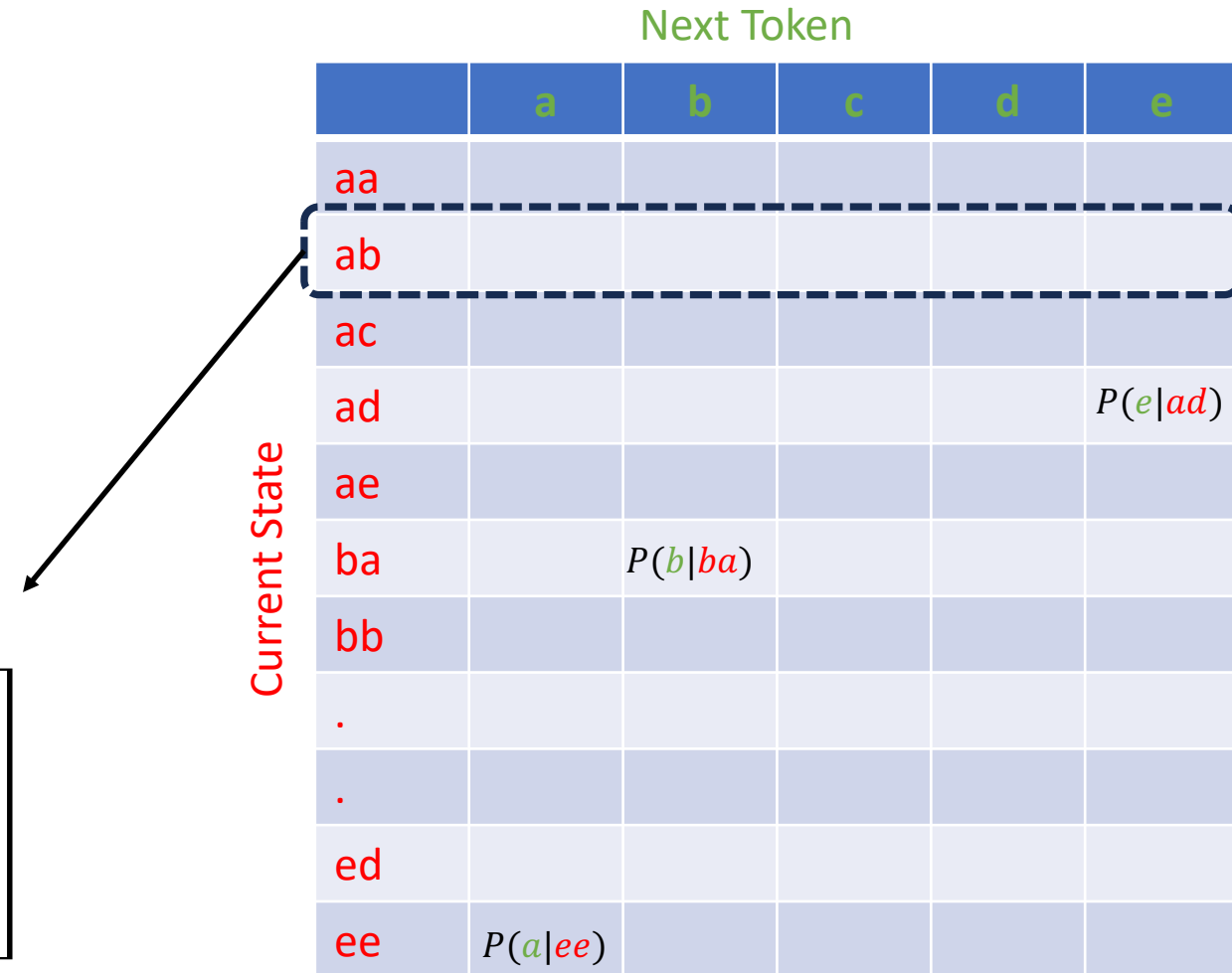
- Assume $V = \{a, b, c, d, e\}$
 - Size of Vocabulary: $|V| = 5$
- Second-order Markov Chains:
 - $n = 2$
 - Number of states: $|V|^2 = 25$
- States:
 - $S = V$
 - $= \{aa, ab, ac, ad, ae, ba, bb, \dots, ea, eb, ec, ed, ee\}$

		Next Token				
		a	b	c	d	e
Current State	aa					
	ab					
	ac					
	ad					$P(e ad)$
	ae					
	ba		$P(b ba)$			
	bb					
	.					
	.					
	ed					
	ee	$P(a ee)$				

Character-level LMs: Text Generation

- Assume $n = 2$
- Initial State: $s_1 = (x_1, x_2)$
- For each time step:
 - Sample next token from $P(. | s_1)$
 - Example:
 - $s_1 = (a, b)$

$$\text{Sample next token from } \mathbf{P}(. | s_1) = \begin{bmatrix} P(a|ab) \\ P(b|ab) \\ P(c|ab) \\ P(d|ab) \\ P(e|ab) \end{bmatrix}$$



		a	b	c	d	e
aa						
ab						
ac						
ad						$P(e ad)$
ae						
ba			$P(b ba)$			
bb						
.						
.						
ed						
ee	$P(a ee)$					

Character-level LMs: Text Generation

- Assume $n = 2$
- Initial State: $s_1 = (x_1, x_2)$
- For each time step:
 - Sample next token from $P(. | s_1)$
 - Example:
 - $s_1 = (a, b)$
 - Sample next token y_1 from $\mathbf{P}(. | s_1)$
 - Let y_1 be the next sample, next state: $s_2 = (b, y_1)$
 - Sample next token y_2 from $\mathbf{P}(. | s_2)$
 - Next state: $s_3 = (y_1, y_2)$
 - Sample next token y_3 from $\mathbf{P}(. | s_3)$
 - And so on...

		Next Token				
		a	b	c	d	e
Current State	aa					
	ab					
	ac					
	ad					$P(e ad)$
	ae					
	ba		$P(b ba)$			
	bb					
	.					
	.					
	ed					
	ee	$P(a ee)$				

Word-level Language Modelling

Word-level LMs

- Vocabulary is the set of all words
 - Eg: $V = \{Markov, Chains, Rock\}$
- First-order MC

		Next Token		
Current State		Markov	Chains	Rock
	Markov			
	Chains			
	Rock			

Word-level LMs

- Vocabulary is the set of all words
 - Eg: $V = \{Markov, Chains, Rock\}$
- Second-order MC:
 - Number of states: $|V|^2 = 9$

		Next Token		
		<i>Markov</i>	<i>Chains</i>	<i>Rock</i>
Current State	<i>(Markov, Markov)</i>			
	<i>(Markov, Chains)</i>			
	<i>(Markov, Rock)</i>			
	<i>(Chains, Markov)</i>			
	<i>(Chains, Chains)</i>			
	<i>(Chains, Rock)</i>			
	<i>(Rock, Markov)</i>			
	<i>(Rock, Chains)</i>			
	<i>(Rock, Rock)</i>			

Word-level LMs: Text Generation

- Initial State: $s_1 = (x_1, x_2)$
- For each time step:
 - Sample next token from $P(. | s_1)$
 - Example:
 - $s_1 = (\text{Markov}, \text{chains})$

- Sample next token from $\mathbf{P}(. | s_1) = \begin{bmatrix} P(\text{Markov} | s_1) \\ P(\text{Chains} | s_1) \\ P(\text{Rock} | s_1) \end{bmatrix}$

		Next Token		
		Markov	Chains	Rock
Current State	(Markov, Markov)			
	(Markov, Chains)			
	(Markov, Rock)			
	(Chains, Markov)			
	(Chains, Chains)			
	(Chains, Rock)			
	(Rock, Markov)			
	(Rock, Chains)			
	(Rock, Rock)			

$$V = \{\text{Markov}, \text{Chains}, \text{Rock}\}$$

How to get Transition Model?

- Maximum likelihood Estimator

$$P_{MLE}(w_t | w_{t-n}, \dots, w_{t-1}) = \frac{C(w_{t-n}, \dots, w_{t-1}, w_t)}{\sum_{w' \in V} C(w_{t-n}, \dots, w_{t-1}, w')}$$

- Example: Markov Chains Rock Markov Rock Chains Rock

$$\bullet P_{MLE}(Chains | Markov) = \frac{C(Markov, Chains)}{C(Markov, Markov) + C(Markov, Chains) + C(Markov, Rock)}$$

$$= \frac{1}{0+1+1} = 0.5$$

$$\bullet P_{MLE}(Rock | Chains) = \frac{C(Chains, Rock)}{C(Chains, Markov) + C(Chains, Chains) + C(Chains, Rock)}$$

$$= \frac{1}{0+1+1} = 0.5$$

PTB Dataset

- PTB: Penn Tree Bank: derived from Wall Street Journal (WSJ)
- Dataset: Train, Test, and Validation
- Heavily sanitized with small fixed vocabulary
- Commonly used for language modelling bench marking

Previewing ptb/train.txt:

```
aer banknote berlitz calloway centrust cluett fromstein gitano guterman hydro-quebec ipo kia memotec mlx nahb punts rake regatta rubens sim snack-food ssangyong swapo wachter
pierre <unk> N years old will join the board as a nonexecutive director nov. N
mr. <unk> is chairman of <unk> n.v. the dutch publishing group
rudolph <unk> N years old and former chairman of consolidated gold fields plc was named a nonexecutive director of this british industrial conglomerate
a form of asbestos once used to make kent cigarette filters has caused a high percentage of cancer deaths among a group of workers exposed to it more than N years ago researchers reported
the asbestos fiber <unk> is unusually <unk> once it enters the <unk> with even brief exposures to it causing symptoms that show up decades later researchers said
<unk> inc. the unit of new york-based <unk> corp. that makes kent cigarettes stopped using <unk> in its <unk> cigarette filters in N
although preliminary findings were reported more than a year ago the latest results appear in today 's new england journal of medicine a forum likely to bring new attention to the problem
a <unk> <unk> said this is an old story
we 're talking about years ago before anyone heard of asbestos having any questionable properties
```

PTB Preprocessing:

- All tokens are lowercased (Eg: That -> that)
- Rare and OOV words are replaced by <unk> token
- All digits are mapped to a single token 'N'
- Tokenization
 - Apostrophes split as separate tokens around possessives and contractions:
 - Eg: "today 's", "we 're".
 - Periods can remain attached to abbreviations:
 - Eg: "mr.", "nov.".
 - Hyphenated compounds retained
 - Eg: "new york-based".

Corpus Class

- Two attributes
 - Dictionary
 - Different from Python's dictionary
 - Dictionary is a class with two attributes
 - word2idx
 - Assigns a unique index for each word
 - Idx2word
 - Provides a reverse mapping from index to word
 - Data
 - Sequence of indices

```
def __init__(self, file_path, file_name):  
    """  
    Initializes the Corpus by creating a dictionary and processing the input file.  
  
    Args:  
        file_path (str): Directory path containing the text files.  
        file_name (str): File name (e.g., 'train.txt').  
    """  
    self.dictionary = Dictionary()  
    full_path = os.path.join(file_path, file_name)  
    self.data = self.tokenize(full_path)
```

```
class Dictionary:  
    """  
    Manages a bidirectional vocabulary mapping between words and unique indices.  
  
    Attributes:  
        word2idx (dict): Maps each word (str) to its unique index (int).  
        idx2word (list): Stores words such that idx2word[index] returns the word.  
  
    Example:  
        >>> dictionary = Dictionary()  
        >>> idx = dictionary.add_word("hello") # adds "hello" and returns its index  
        >>> print(dictionary.word2idx["hello"]) # prints the index for "hello"  
        >>> print(dictionary.idx2word[idx]) # prints "hello"  
        >>> print(len(dictionary)) # prints vocabulary size (e.g., 1)  
    """  
  
    def __init__(self):  
        """  
        Initializes an empty Dictionary.  
        """  
        self.word2idx = {}  
        self.idx2word = []
```

Summary of PTB Dataset

```
print("Basic Corpus Information:")
print(f"Vocabulary size: {len(corpus.dictionary)}")
print(f"Number of tokens in training data: {len(corpus.data)}")

# Look at the first few tokens and their indices
n = 5
print(f"\nFirst {n} tokens and their indices:")
first_n_indices = corpus.data[:n]
for idx in first_n_indices:
    word = corpus.dictionary.idx2word[idx]
    print(f"Index: {idx:4d}, Word: {word}")

# Get some statistics about the vocabulary
print("\nVocabulary Examples:")
print("First 10 words in vocabulary:", corpus.dictionary.idx2word[:10])
print("Last 10 words in vocabulary:", corpus.dictionary.idx2word[-10:])

# Look up some specific words
sample_words = ['the', 'a', '<eos>', '<unk>']
print("\nIndices for common words:")
for word in sample_words:
    if word in corpus.dictionary.word2idx:
        print(f"'{word}': {corpus.dictionary.word2idx[word]}")
    else:
        print(f"'{word}' not in vocabulary")

# Convert a small sequence back to words
print("\nSample sequence converted back to words:")
sample_sequence = corpus.data[100:110] # Get 10 tokens
reconstructed_text = ' '.join([corpus.dictionary.idx2word[idx] for idx in sample_sequence])
print(reconstructed_text)

# Get some basic statistics
unique_indices = len(set(corpus.data))
print(f"\nNumber of unique tokens used in training data: {unique_indices}")
print(f"Total vocabulary size: {len(corpus.dictionary)}")
```

Estimation of Transition Probabilities

- Transition Probabilities