UNIT 2----

Unsmoothed N-grams and Their Evaluation

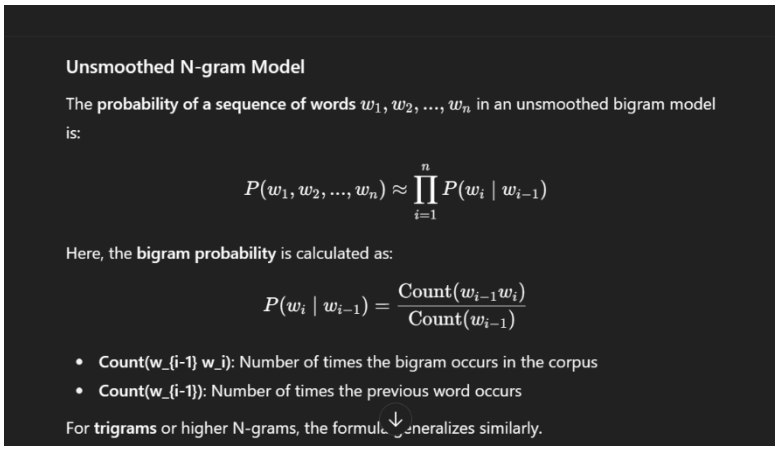
Introduction

N-grams are contiguous sequences of N items from a given text or speech sequence. In NLP, these items are usually words or characters. An unsmoothed N-gram model estimates the probability of a word sequence solely based on observed counts in the training corpus, without applying any correction for unseen events

Example

Suppose we have a corpus:
"I like NLP", "I like AI"

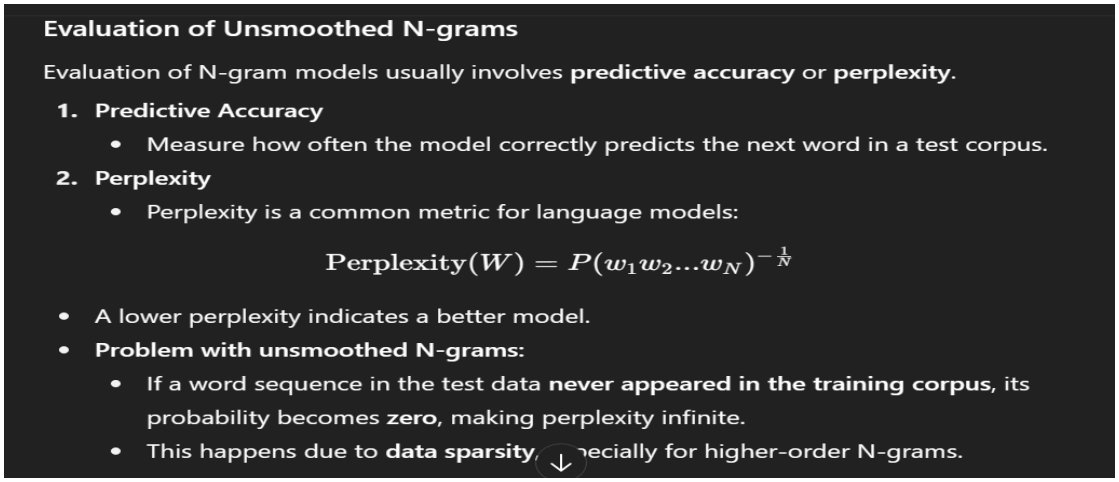
- Bigram counts:
    - Count(I like) = 2
    - Count(like NLP) = 1
    - Count(like AI) = 1



- Bigram probability without smoothing:
    - P(NLP | like) = 1/2
    - P(ML | like) = 0 → Zero probability for unseen bigrams.

Significance

- Unsmoothed N-grams are simple and easy to compute.
- They **capture word dependencies** in local contexts.
- However, they **fail to handle unseen sequences**, motivating the need for **smoothing techniques**.



2. Smoothing in N-grams: Interpolation and Backoff Techniques

Introduction

In N-gram models, data sparsity is a major problem: many word sequences in the test corpus may never appear in the training corpus, leading to zero probability estimates. Smoothing is a set of techniques to adjust these probabilities to account for unseen events and provide non-zero probabilities for all possible word sequences.

What is Smoothing?

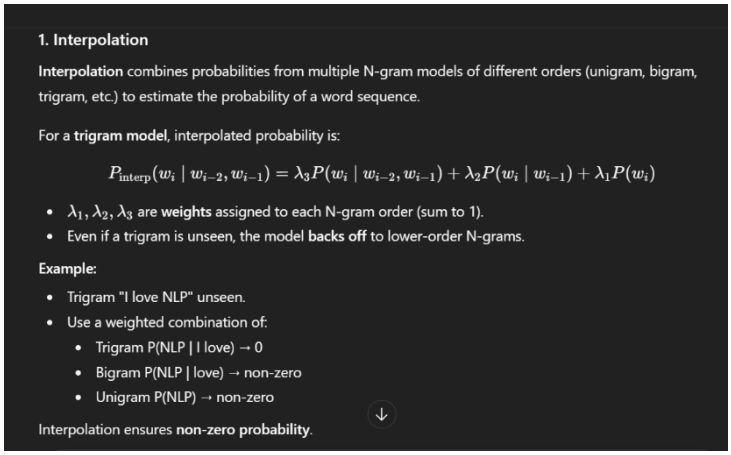
Smoothing modifies the raw probability estimates from N-gram counts to:

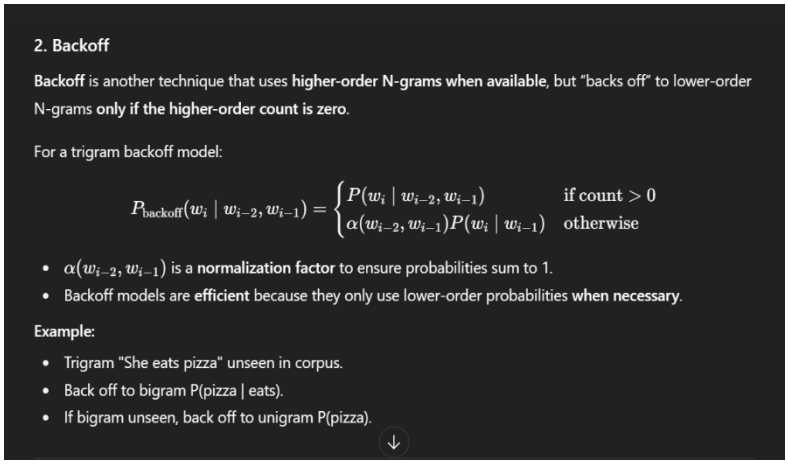
Avoid zero probabilities for unseen sequences.

Redistribute some probability mass from seen events to unseen events.

Improve generalization on unseen data.

Common smoothing methods include Laplace/Add-one smoothing, Good-Turing, interpolation, and backoff.





Significance of Smoothing

Prevents zero probabilities for unseen sequences.

Improves robustness of language models.

Essential for applications like speech recognition, machine translation, and text generation.

3. Different Word Classes and Their Significance in NLP

Major Word Classes

1. **Nouns (N)**
    - **Represent people, places, things, or concepts.**
    - **Examples: *student, city, happiness***

- **Significance: Identify subjects and objects in sentences; crucial for information extraction.**

2. **Pronouns (PRP)**
    - **Substitute for nouns to avoid repetition.**
    - **Examples: *he, she, it, they***
    - **Significance: Important for coreference resolution and anaphora resolution in NLP.**
3. **Verbs (V)**
    - **Denote actions, occurrences, or states.**
    - **Examples: *run, eat, is, seem***
    - **Significance: Critical for syntactic parsing, sentence understanding, and event extraction.**
4. **Adjectives (ADJ)**
    - **Describe qualities or attributes of nouns.**
    - **Examples: *beautiful, fast, intelligent***
    - **Significance: Useful in sentiment analysis, text summarization, and question answering.**
5. **Adverbs (ADV)**
    - **Modify verbs, adjectives, or other adverbs.**
    - **Examples: *quickly, very, almost***
    - **Significance: Enhance understanding of actions, intensity, and manner in NLP tasks.**
6. **Prepositions (PREP)**
    - **Show relation between nouns/pronouns and other words.**
    - **Examples: *in, on, at, under***
    - **Significance: Crucial for dependency parsing and spatial/temporal reasoning.**
7. **Conjunctions (CONJ)**
    - **Connect words, phrases, or clauses.**
    - **Examples: *and, but, because***

- **Significance: Help analyze sentence structure and coordination in text.**

8. **Determiners (DET)**
    - **Introduce nouns and specify reference.**
    - **Examples: *a, an, the, this***
    - **Significance: Important for noun phrase recognition and syntactic parsing.**
9. **Interjections (INTJ)**
    - **Express emotion or reaction.**
    - **Examples: *oh!, wow!, alas!***
    - **Significance: Useful in sentiment analysis and dialogue systems.**

Significance of Word Classes in NLP

1. **Part-of-Speech Tagging (PoS Tagging)**
    - **Identifying word classes is the first step in PoS tagging, which is essential for syntactic analysis.**
2. **Syntactic Parsing**
    - **Helps in building parse trees for understanding sentence structure.**
3. **Named Entity Recognition (NER)**
    - **Differentiating nouns from other word classes helps identify entities like people, organizations, and locations.**
4. **Machine Translation & Text Generation**
    - **Correct word class identification ensures grammatically correct translations and generated text.**
5. **Sentiment Analysis**
    - **Adjectives and adverbs play a key role in detecting sentiment or opinion.**

4. Part-of-Speech (PoS) Tagging and Its Importance

Part-of-Speech (PoS) Tagging is the process of assigning each word in a sentence its grammatical category (such as noun, verb, adjective, etc.) based on both its definition and context.

It is a fundamental task in Natural Language Processing (NLP) because it provides syntactic information about words in a text.

PoS tagging involves labeling words with **tags** that represent their **word class**.

Example:

Sentence: *"The cat sleeps on the mat."*
PoS Tags:

- The → Determiner (DET)
- cat → Noun (N)
- sleeps → Verb (V)
- on → Preposition (PREP)
- the → Determiner (DET)
- mat → Noun (N)

Techniques for PoS Tagging

PoS tagging can be performed using:

1. **Rule-based approaches** – Use manually crafted grammatical rules.
2. **Stochastic approaches** – Use probability models based on word sequences.
3. **Transformation-based approaches** – Combine rules and statistical learning.

Importance of PoS Tagging in NLP

1. **Syntactic Analysis**
    - PoS tags help in **parsing sentences** and building **syntax trees**, enabling machines to understand sentence structure.
2. **Word Sense Disambiguation (WSD)**
    - Words with multiple meanings can be distinguished based on their PoS tag.
    - Example: *“Book a ticket”* (Verb) vs *“Read a book”* (Noun)
3. **Named Entity Recognition (NER)**
    - Correct identification of nouns and proper nouns is essential for **extracting entities** like people, places, and organizations.

4. **Information Retrieval and Extraction**
    - PoS tagging helps systems **identify key phrases** and **extract relevant information** from text.
5. **Machine Translation & Text Generation**
    - Accurate PoS tags ensure **grammatically correct translations** and generated sentences.
6. **Sentiment Analysis**
    - Identifying adjectives, adverbs, and verbs is important for **understanding opinions and emotions**.

Challenges in PoS Tagging

- Ambiguity: Words can belong to **multiple word classes** depending on context.
- Data sparsity: Rare words may not appear in the training corpus.
- Complex grammar: Some languages have **rich inflections**, making tagging more difficult.

5. **Comparison of Rule-based, Stochastic, and Transformation-based PoS Tagging Approaches**

Introduction

Part-of-Speech (PoS) tagging can be implemented using different approaches. The **choice of approach** affects **accuracy, efficiency, and adaptability** of NLP systems. The three main approaches are:

1. **Rule-based Tagging**
2. **Stochastic (Statistical) Tagging**
3. **Transformation-based Tagging**

---

1. Rule-based PoS Tagging

Definition:

- Uses **manually created linguistic rules** to assign PoS tags to words.
- Relies on dictionaries and grammar rules.

Example Rules:

- If a word ends in “-ly”, tag as **adverb**.
- If a word follows a determiner, tag as **noun**.

Advantages:

- Transparent and **interpretable**.
- Works well for **well-defined grammatical structures**.

Disadvantages:

- **Time-consuming** to create rules.
- Cannot handle all **exceptions and ambiguities**.
- Not scalable.

---

2. Stochastic (Statistical) PoS Tagging

Definition:

- Uses **probabilities** of word sequences from a **training corpus** to assign tags.
- Commonly implemented using **Hidden Markov Models (HMMs)**.

Example:

- P(tag | word) × P(tag | previous tag) is used to predict the most probable tag.

Advantages:

- Can handle **ambiguity** using context.
- Scalable.
- Adapts automatically to new data if trained properly.

Disadvantages:

- Requires a **large annotated corpus** for training.
- May produce errors for **rare words or unseen sequences**.

---

3. Transformation-based PoS Tagging

Definition:

- Also called **Brill Tagging**.

- Starts with a **baseline tagging** (e.g., assigning most frequent tag for each word) and **iteratively applies transformation rules** to correct errors.

Example Transformation Rule:

- If a word is tagged as a noun but follows "to", change it to a verb.

Advantages:

- Combines **rule-based interpretability** and **statistical learning**.
- High accuracy without requiring very large corpora.

Disadvantages:

- Rule learning can be **computationally intensive**.
- May still struggle with highly ambiguous contexts.

6. **Issues in Part-of-Speech (PoS) Tagging**

Introduction

Part-of-Speech (PoS) tagging is a fundamental NLP task, but it faces several **challenges and issues** due to the complexity of natural languages. Understanding these issues is essential for designing **robust tagging systems**.

---

1. Ambiguity

- **Definition:** A word may have **multiple possible PoS tags** depending on context.
- **Example:**
  - *"Book a ticket"* → *Book* = Verb
  - *"Read a book"* → *Book* = Noun
- **Impact:** Ambiguity is the **primary source of errors** in PoS tagging.

---

2. Unknown Words (Out-of-Vocabulary Words)

- Words not present in the training corpus may appear in the test corpus.
- **Example:** Technical terms, slang, or new words (*cryptocurrency*).
- **Impact:** Statistical taggers assign **low or zero probability**, leading to misclassification.

---

3. Morphological Complexity

- Languages with **rich morphology** (e.g., Telugu, Finnish, Arabic) have multiple forms of a single word.
- **Example:** In English, “walks”, “walked”, “walking” → all derived from *walk*.
- **Impact:** Increased difficulty in correctly assigning tags without **lemmatization or morphological analysis**.

---

4. Sparse Data Problem

- High-order N-grams for statistical PoS tagging may be **rare or absent** in the training data.
- **Impact:** Probability estimation becomes unreliable for unseen sequences.

---

5. Context Sensitivity

- The correct tag often depends on **long-range context**, not just immediate neighbors.
- **Example:**
    - *“I saw her duck”* → *duck* can be Noun or Verb
- **Impact:** Simple models using limited context may fail.

---

6. Inconsistent Tagging Standards

- Different corpora may use **different tag sets** or annotation schemes.
- **Impact:** Models trained on one corpus may not perform well on another.

---

7. Multi-word Expressions

- Certain phrases function as a **single syntactic unit** but consist of multiple words.
- **Example:** *“Kick the bucket”* → Idiomatic meaning is lost if words are tagged individually.
- **Impact:** PoS tagging may fail to capture **idiomatic or phrasal structures**.

---

8. Errors in Training Data

- Statistical and transformation-based taggers depend on **annotated corpora**.

- Mistakes in the training data can **propagate errors** to the model.

7. Hidden Markov Models (HMM) and Maximum Entropy Models for PoS Tagging

Introduction

Part-of-Speech (PoS) tagging is a sequential labeling task where each word in a sentence is assigned a grammatical tag. Two widely used **statistical models** for PoS tagging are:

1. **Hidden Markov Models (HMMs)**
2. **Maximum Entropy Models (MEMs)**

Both provide probabilistic frameworks for predicting the most likely tags given a sequence of words.

A. Hidden Markov Models (HMM) for PoS Tagging

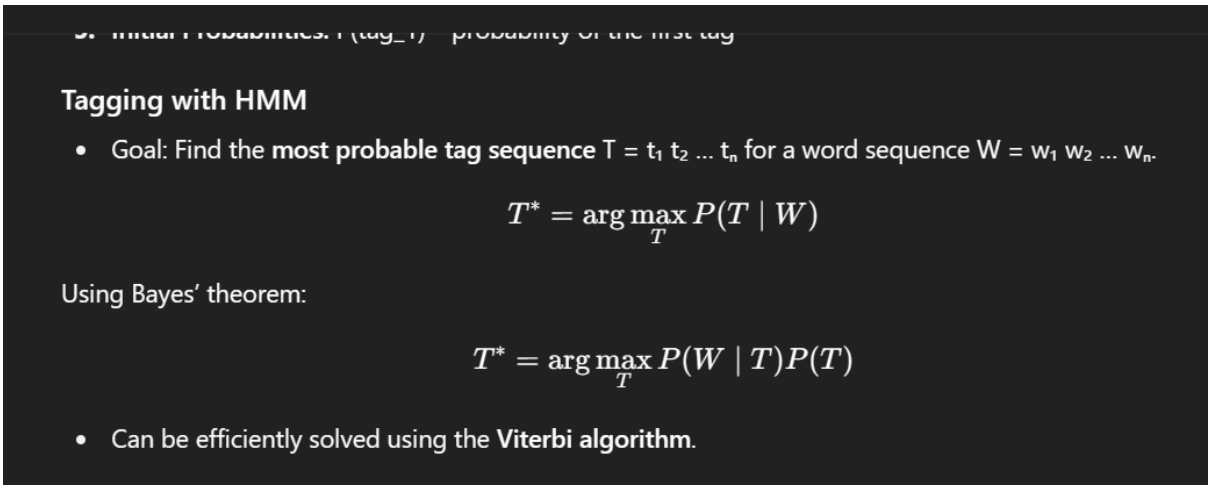
Definition

- HMM is a **generative probabilistic model**.
- Assumes that the sequence of tags forms a **Markov chain**, where the current tag depends only on the **previous tag(s)**.
- Observed words are **emissions** from these hidden tags.

Components of HMM

1. **States (Tags):** PoS tags (Noun, Verb, etc.)
2. **Observations (Words):** Words in the sentence
3. **Transition Probabilities:** P(tag_i | tag_{i-1}) – probability of tag given the previous tag
4. **Emission Probabilities:** P(word_i | tag_i) – probability of a word given a tag

5. **Initial Probabilities:** P(tag_1) – probability of the first tag



Advantages

- Handles **contextual dependencies** via transition probabilities
- Works well for **large corpora**
- Well-established in NLP

Disadvantages

- Assumes **Markov property** (limited context)
- Requires **large annotated corpus** for accurate probabilities
- Struggles with **rare/unknown words** without smoothing

B. Maximum Entropy Models (MEMs) for PoS Tagging

Definition

- MEMs are **discriminative probabilistic models**.
- Directly model **P(tag | context)** using **features from the data**, without assuming independence between observations.
- Use **entropy maximization** to estimate probabilities that **best match the observed features**.

Features Used

- Current word
- Prefixes/suffixes
- Surrounding words

- Capitalization, hyphenation, numeric forms
- Previous tags (optional)

Probability Formula

$$P(t \mid context) = \frac{1}{Z(context)} \exp\left(\sum_i \lambda_i f_i(t, context)\right)$$

- f_i = feature functions
- λ_i = learned weights
- $Z(context)$ = normalization factor

Advantages

- Can incorporate **arbitrary and overlapping features**
- Better handles **unknown or rare words**
- Discriminative: focuses on **P(tag | word sequence)** rather than modeling joint probability

Disadvantages

- Computationally more intensive
- Requires **feature engineering** for best performance
- Training can be slower compared to HMM

Comparison: HMM vs MEM

Feature	HMM	Maximum Entropy
Model Type	Generative	Discriminative
Probability Modeled	P(words, tags)	P(tag
Context Handling	Limited (Markov assumption)	Flexible (any features)
Unknown Words	Needs smoothing	Can handle better with features

Feature	HMM	Maximum Entropy
Computation	Efficient (Viterbi)	Slower training
Accuracy	Moderate	Often higher with good features
