Concepts:

**Language Model (LM) Overview**

* **Definition**: Probability distribution over a sequence of tokens (characters, words, subwords, numbers).
* **Objective**: Predict the next token given a sequence.

**Statistical Language Model**

* **Origin (1950s)**: Claude Shannon introduced the character guessing game.
* **Entropy & Perplexity**:
  + Entropy: Measure of randomness in a language.
  + Perplexity: Evaluates LM performance; linked to entropy.
* **Real-world Applications**:
  + Auto-complete (Google Search, keyboards).
  + Speech recognition (error correction in text from speech).
  + Machine translation (correct word choice in context).
  + Spell checking, text generation, and more.

**Mathematical Foundation**

* **Chain Rule Decomposition**:
  + P(W1, W2, ..., Wn) = P(W1) \* P(W2 | W1) \* P(W3 | W1, W2) \* ...
* **Conditional Probabilities**:
  + Estimated using corpus frequency: P(Wn | W1, ..., Wn-1) = Count(W1, ..., Wn) / Count(W1, ..., Wn-1)
  + Issue: As context size increases, probability estimation becomes sparse.

**Markov Assumption & N-Gram Models**

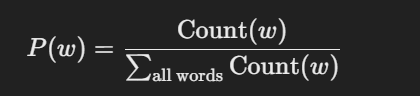
* **Markov Assumption**: The probability of a word depends only on a fixed number of previous words.
* **N-Gram Models**:
  + **Unigram (0th-order Markov Model)**: P(Wn) (Each word independent)
  + **Bigram (1st-order Markov Model)**: P(Wn | Wn-1)
  + **Trigram (2nd-order Markov Model)**: P(Wn | Wn-2, Wn-1)
  + Higher n-grams consider longer context.
* **Advantage**: Reduces sparsity by limiting context length.

**Unigram and Bigram Count Table**

The **unigram and bigram count table** is used in **language modeling** to analyze word frequencies and transitions in a corpus.

**1. Unigram Count Table**

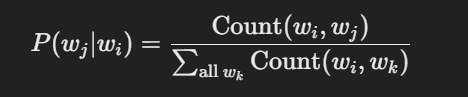
* **Records individual word occurrences** in the corpus.
* Used to compute **unigram probabilities**:



* **Does not capture word order**—only gives independent word frequencies.

**2. Bigram Count Table**

* **Records word pair (bigram) occurrences** where one word follows another.
* Used to compute **bigram probabilities**:



* Helps in **predictive text, speech recognition, and NLP tasks**.

**Key Properties**

1. **Bigrams depend on sequence**: Unlike unigrams, bigram counts tell us how words follow each other.
2. **Sum of row entries = sum of column entries**: Since each word that appears as a second word in a bigram must have appeared as the first word in another bigram, the total counts match.
3. **Cannot derive bigram counts from unigram counts**: Since unigram counts do not capture order, bigram counts need direct observation from the corpus.
4. **Unigram and bigram probabilities** are useful in **language models** like **n-gram models, Hidden Markov Models, and deep learning NLP models**.

**Bigram probability Matrix:**

**1. Sparsity of the Bigram Matrix**

* For a **large vocabulary (e.g., 1 million words)**, the bigram matrix is **mostly zero**.
* In the **BLE Restaurant Project**, **99.95%** of bigram entries were zero.

**3. Meaning of Zero Entries**

* **Zero entries = word pairs that never appeared in the training corpus.**
* **Not all zero entries are invalid:**
  + Some **zeroes are due to data sparsity** (e.g., "enjoyed the festival" may not have appeared in training but is valid).
  + Others **are grammatically or semantically impossible** (e.g., "each want" or "food want" are incorrect).

**4. Insights from the Bigram Table**

* **Common bigrams indicate knowledge types:**
  + High probability bigrams (e.g., "Chinese food") suggest **world knowledge**.
  + Certain patterns (e.g., "one to") indicate **grammatical structures**.

**5. Contingent Zero Concept**

* **Contingent zeros:** Entries that are hard to construct valid sentences for.
  + Example: **"two food"** rarely appears, but valid sentences exist ("I can't express my love to food").
  + These zeros are different from impossible bigrams like **"each want"**, which are ungrammatical.

**1. Limitations of N-gram Models**

* **Long-range dependencies:** N-gram models struggle with capturing long-term dependencies (e.g., in “The project which he had been working on for months was finally approved by the committee,” the word “approved” refers to “project,” which is far apart).
* **Fixed context size:** Trigram or 4-gram models fail to predict words when the context required is longer than their fixed window.
* **Count-zero problem:** Unseen word sequences in test data receive a probability of zero.
* **Handling long-range dependencies:** Models like **RNNs, LSTMs, and GRUs** address this issue better.

**2. Maximum Likelihood Estimation (MLE) Issues**

* **Probability estimation using frequency counts**:

P(wn∣wn−1,wn−2)=Count(wn−2,wn−1,wn)Count(wn−2,wn−1)P(w\_n | w\_{n-1}, w\_{n-2}) = \frac{\text{Count}(w\_{n-2}, w\_{n-1}, w\_n)}{\text{Count}(w\_{n-2}, w\_{n-1})}P(wn​∣wn−1​,wn−2​)=Count(wn−2​,wn−1​)Count(wn−2​,wn−1​,wn​)​

* **Zero probability issue:** If a valid word sequence is missing in training data, MLE assigns zero probability, making the model unreliable.

**3. Out-of-Vocabulary (OOV) Words**

* **OOV problem:** Words in the test set that never appeared in training cause issues (e.g., "LOL").
* **Handling OOV words:**
  + Use context-based prediction to infer meaning.
  + Maintain a **lexicon** (subset of vocabulary with frequent words).
  + Use a **UNK (unknown token)** for words below a frequency threshold (e.g., words appearing <5 times are replaced with "UNK").

**4. Smoothing Techniques**

To handle sparse matrices (where many probabilities are zero), we use smoothing techniques:

**a. Add-One Smoothing (Laplace Smoothing)**

* Adds **1** to all N-gram counts:

P(wn∣wn−1)=Count(wn−1,wn)+1Count(wn−1)+VP(w\_n | w\_{n-1}) = \frac{\text{Count}(w\_{n-1}, w\_n) + 1}{\text{Count}(w\_{n-1}) + V}P(wn​∣wn−1​)=Count(wn−1​)+VCount(wn−1​,wn​)+1​

where **V** is the vocabulary size.

* **Issue:** Overestimates low-frequency words.

**b. Add-K Smoothing (Generalization of Add-One)**

* Adds **K** instead of **1** to counts:

P(wn∣wn−1)=Count(wn−1,wn)+KCount(wn−1)+KVP(w\_n | w\_{n-1}) = \frac{\text{Count}(w\_{n-1}, w\_n) + K}{\text{Count}(w\_{n-1}) + K V}P(wn​∣wn−1​)=Count(wn−1​)+KVCount(wn−1​,wn​)+K​

* More flexible than add-one smoothing.

c. **. Unigram Prior Smoothing**

* **Basic idea:** Assign a small probability to all words based on their **unigram frequency**.
* Formula:

P(wn∣wn−1)=C(wn−1,wn)+βP(wn)C(wn−1)+βP(w\_n | w\_{n-1}) = \frac{C(w\_{n-1}, w\_n) + \beta P(w\_n)}{C(w\_{n-1}) + \beta}P(wn​∣wn−1​)=C(wn−1​)+βC(wn−1​,wn​)+βP(wn​)​

* + **β (smoothing factor)** determines how much unigram probabilities influence the estimate.
  + **Ensures** that all words have **some probability**, even if unseen in bigram/trigram training data.

**5. Impact of Smoothing**

* **Eliminates zero probabilities** but alters frequency distributions.
* High-frequency words lose probability while low-frequency words gain some.

**5. Advanced Smoothing Techniques**

**5.1. Backoff Smoothing**

* If **higher-order n-grams (trigrams/bigrams)** have **zero probability**, the model "backs off" to a **lower-order n-gram (bigram/unigram)**.
* **Example:**
  + If **P(“approved” | “was finally”)** is **0**, use **P(“approved” | “finally”)** instead.
  + Formula:

Pbackoff(wn∣wn−1)=αP(wn)P\_{backoff}(w\_n | w\_{n-1}) = \alpha P(w\_n)Pbackoff​(wn​∣wn−1​)=αP(wn​)

* + **α (scaling factor)** ensures probability distribution remains valid.

**5.2. Interpolation Smoothing**

* Unlike **Backoff**, which discards higher-order models when they fail, **Interpolation** **combines** multiple n-gram models.
* Formula:

P(wn∣wn−2,wn−1)=λ3P(wn∣wn−2,wn−1)+λ2P(wn∣wn−1)+λ1P(wn)P(w\_n | w\_{n-2}, w\_{n-1}) = \lambda\_3 P(w\_n | w\_{n-2}, w\_{n-1}) + \lambda\_2 P(w\_n | w\_{n-1}) + \lambda\_1 P(w\_n)P(wn​∣wn−2​,wn−1​)=λ3​P(wn​∣wn−2​,wn−1​)+λ2​P(wn​∣wn−1​)+λ1​P(wn​)

* + **λ values** are weights that sum to 1.
  + Ensures **lower-order n-grams still contribute**, even if higher-order models have nonzero probability.

**Advanced Smoothing Techniques**

These techniques address the problems faced in traditional smoothing methods.

**1. Good-Turing Smoothing**

* **Intuition:** Use the count of things seen once to estimate the count of things never seen.
* **Frequency of Frequencies (NC):** Count how many times each count appears in the corpus.
  + Example:
    - Words: {“I” (3), “Rohan” (2), “like” (1), etc.}
    - **N1** (count of words occurring once) = 3
    - **N2** (count of words occurring twice) = 2
    - **N3** (count of words occurring thrice) = 1
* **Probability Estimation:**
  + If **MLE** says the probability of an unseen event is 0, Good-Turing modifies it.
  + **For unseen words:** P=N1NP = \frac{N1}{N}P=NN1​
  + **For seen words:** C∗=(C+1)×NC+1NCC^\* = (C+1) \times \frac{N\_{C+1}}{N\_C}C∗=(C+1)×NC​NC+1​​
  + Example:
    - Observed 18 birds.
    - **MLE:** P(woodpecker)=118P(\text{woodpecker}) = \frac{1}{18}P(woodpecker)=181​.
    - **Good-Turing:** P(new bird)=N118=318P(\text{new bird}) = \frac{N1}{18} = \frac{3}{18}P(new bird)=18N1​=183​.
    - This ensures a nonzero probability for unseen words while slightly adjusting existing word probabilities.

**2. Absolute Discounting Interpolation**

* **Observation from Good-Turing:** The modified count is approximately the original count minus a constant (e.g., 0.75).
* **Idea:** Instead of complex computation, subtract a fixed discount **D** from the count.
* **Formula:**

P(Wi∣Wi−1)=C(Wi−1,Wi)−DC(Wi−1)+λP(Wi)P(W\_i | W\_{i-1}) = \frac{C(W\_{i-1}, W\_i) - D}{C(W\_{i-1})} + \lambda P(W\_i)P(Wi​∣Wi−1​)=C(Wi−1​)C(Wi−1​,Wi​)−D​+λP(Wi​)

* + **First term:** Adjusted probability for bigram.
  + **Second term:** **Interpolation** with unigram probability (to handle low-count cases).
  + **Lambda (λ\lambdaλ)**: Ensures probability mass consistency.

**3. Continuation Probability of a Unigram**

* Instead of using **unigram probability**, we measure **continuation probability**.
* **Unigram probability problem:** Frequent words (e.g., "Angeles") get high probability due to high frequency, even if they rarely appear in diverse contexts.
* **Continuation probability:** Measures how many unique **bigrams** a unigram completes.

**Intuition Behind Continuation Probability**

* Example: "My breakfast is incomplete without a cup of \_\_\_."
  + **Unigram probability**: Might favor “Angeles” due to high frequency.
  + **Continuation probability**: “Coffee” appears in **many** bigrams (hot coffee, morning coffee, etc.), so it gets a higher score.

**Formula for Continuation Probability**

Pcont(w)=Number of unique bigrams completed by wTotal number of unique bigrams in the corpusP\_{cont}(w) = \frac{\text{Number of unique bigrams completed by } w}{\text{Total number of unique bigrams in the corpus}}Pcont​(w)=Total number of unique bigrams in the corpusNumber of unique bigrams completed by w​

* **Numerator:** Unique bigrams where **w appears as the second word**.
* **Denominator:** Total unique bigrams in the corpus.

**4. Absolute Discounting in Smoothing**

* We use **absolute discounting (D)** to adjust probabilities.
* Formula:

P(wi∣wi−1)=max⁡(Count(wi−1,wi)−D,0)/Count(wi−1)+λPcont(wi)P(w\_i | w\_{i-1}) = \max(\text{Count}(w\_{i-1}, w\_i) - D, 0) / \text{Count}(w\_{i-1}) + \lambda P\_{cont}(w\_i)P(wi​∣wi−1​)=max(Count(wi−1​,wi​)−D,0)/Count(wi−1​)+λPcont​(wi​)

* **λ (Lambda) adjustment:** Ensures total probability remains **1**.

λ=D×Number of discounted countsTotal count of bigram row\lambda = \frac{D \times \text{Number of discounted counts}}{\text{Total count of bigram row}}λ=Total count of bigram rowD×Number of discounted counts​

**5. Kneser-Ney Smoothing**

* + Uses **continuation probability** instead of unigram probability.
  + Handles **contextual word usage better**.

**Three Main Smoothing Techniques**

1. **Add-One Smoothing (Laplace Smoothing)**
   * Adds **+1 to all counts** to avoid zero probabilities.
   * Problem: Overestimates rare words.
2. **Good-Turing Smoothing**
   * Adjusts probabilities based on **count frequency**.
   * Example: If unseen bigrams occur **zero times**, they get an estimated probability based on rare bigrams.
3. **Kneser-Ney Smoothing**
   * Uses **continuation probability** instead of unigram probability.
   * Handles **contextual word usage better**.

Summary: