MODULE 2 • CS24351

Module 2: Language Modeling, Smoothing & POS Tagging

Topics 13 to 24 • N-grams, Perplexity, Laplace, Kneser-Ney, HMMs, Viterbi Decoding & CRFs

Module 2: N-gram Language Models, Smoothing Techniques & POS Tagging

📌 **Syllabus & Pedagogical Blueprint (Topics 13 to 24)**

Probabilistic Language Modeling • Chain Rule of Probability & Markov Independence Assumptions • N-gram Language Models (Unigram, Bigram, Trigram) • Maximum Likelihood Estimation (MLE) • The Zero-Probability Sparsity Catastrophe • Perplexity and Cross-Entropy Evaluation • Smoothing Formalisms: Laplace (Add-1), Add-k, Absolute Discounting, Jelinek-Mercer Linear Interpolation, Katz Backoff with Good-Turing Discounting, Kneser-Ney Smoothing (Interpolated & Modified) • Part-of-Speech (POS) Tagging • The Penn Treebank Tagset • Hidden Markov Models (HMM) for POS Tagging: Transition (A), Emission (B), and Initial (π) Distributions • The Viterbi Decoding Algorithm Trellis Trace • Forward-Backward Algorithm • Maximum Entropy Markov Models (MEMM) & The Label Bias Problem • Linear-Chain Conditional Random Fields (CRF) • 15 Solved University Examination Questions.

Topic 13: Probabilistic Language Models & The Markov Chain

A **Language Model (LM)** computes the joint probability distribution $P(W) = P(w_1, w_2, \ldots, w_N)$ over word sequences, or computes the conditional probability of a next word $P(w_N \mid w_1, \ldots, w_{N-1})$.

📐 **The Chain Rule of Probability & The Markov Assumption**

By exact probabilistic chain rule decomposition:

$$\mathbf{P(w_1, w_2, \ldots, w_N) = \prod_{k=1}^{N} P(w_k \mid w_1, w_2, \ldots, w_{k-1})}$$

Because conditioning on the entire unbounded history is computationally intractable, the **k-th Order Markov Assumption** approximates history using only the preceding $n-1$ words:

- **Unigram Model ($n=1$):** $P(w_1, \ldots, w_N) \approx \prod_{k=1}^{N} P(w_k)$ (Complete statistical independence).
- **Bigram Model ($n=2$):** $P(w_1, \ldots, w_N) \approx \prod_{k=1}^{N} P(w_k \mid w_{k-1})$.
- **Trigram Model ($n=3$):** $P(w_1, \ldots, w_N) \approx \prod_{k=1}^{N} P(w_k \mid w_{k-2}, w_{k-1})$.

Topic 14: Maximum Likelihood Estimation (MLE) & Perplexity

In standard MLE, bigram probabilities are estimated via relative corpus frequencies:

$$\mathbf{P_{MLE}(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i)}{\sum_w C(w_{i-1}, w)} = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}}$$

📊 Evaluating Language Models: Perplexity Metric

Perplexity (PP) measures how surprised the language model is by an unseen test set $W = (w_1, \ldots, w_N)$. A lower perplexity indicates a superior predictive model:

$$\mathrm{PP}(\mathbf{W}) = \mathbf{P}(\mathbf{w_1}, \mathbf{w_2}, \ldots, \mathbf{w_N})^{-\frac{1}{N}} = \sqrt[N]{\frac{\mathbf{1}}{\prod_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{P}(\mathbf{w_i} \mid \mathbf{w_{i-1}})}} = \mathbf{2^{H(W)}}$$

$$\text{where Cross-Entropy } H(W) = -\frac{1}{N} \sum_{i=1}^{N} \log_2 P(w_i \mid w_{i-1})$$

Topic 15: Comprehensive Smoothing Techniques

If an unseen bigram appears in test data ($C(w_{i-1}, w_i) = 0$), $P_{\text{MLE}} = 0$, causing the entire corpus probability to crash to zero and $\mathrm{PP}(W) \to \infty$. **Smoothing** reallocates probability mass from frequent n-grams to zero-count events:

Smoothing Technique	Mathematical Formulation	Key Analytical Properties
Laplace (Add-1)	$P_{\text{Laplace}}(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i) + 1}{C(w_{i-1}) + \lvert V \rvert}$	Assigns far too much probability mass to zero counts when vocabulary $\lvert V \rvert$ is large.
Add-k (Lidstone)	$P_{\text{Lidstone}}(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i) + k}{C(w_{i-1}) + k\lvert V \rvert} \quad (0 < k < 1)$	Requires optimizing hyperparameter k on held-out validation data.
Jelinek-Mercer Interpolation	$P_{\text{JM}}(w_i \mid w_{i-1}) = \lambda P_{\text{MLE}}(w_i \mid w_{i-1}) + (1 - \lambda) P_{\text{MLE}}(w_i)$	Linear combination of higher and lower-order models; λ learned via EM.
Kneser-Ney (Interpolated)	$P_{\text{KN}}(w_i \mid w_{i-1}) = \frac{\max(C(w_{i-1}, w_i) - d, 0)}{C(w_{i-1})} + \lambda(w_{i-1}) P_{\text{cont}}(w_i)$	Uses **Continuation Probability** $P_{\text{cont}}(w_i) = \frac{\lvert\{w_{i-1} : C(w_{i-1}, w_i) > 0\}\rvert}{\sum_w \lvert\{w' : C(w', w) > 0\}\rvert}$. State of the art!

🏛️ Solved Numerical 3: Kneser-Ney vs Unigram Probability Calculation

Consider the word *"Francisco"* in a corpus. It occurs 1000 times, but *always and only* immediately following the word *"San"* ($C(\text{San}, \text{Francisco}) = 1000$). Total bigram tokens = 100,000.

- **Standard MLE Unigram Probability:** $P_{\text{MLE}}(\text{Francisco}) = \frac{1000}{100000} = \mathbf{0.01}$ (High unigram probability!). If an unseen prefix like *"delicious"* occurs, a backoff model using unigram MLE would predict *"delicious Francisco"* with high probability!
- **Kneser-Ney Continuation Probability:** How many distinct preceding words does *"Francisco"* complete? Only 1 word (`San`)! Total number of distinct word pairs in vocabulary $= 50,000$.

$$\mathbf{P}_{\text{cont}}(\text{Francisco}) = \frac{\lvert\{\mathbf{w'} : \mathbf{C(w', \text{Francisco}) > 0}\}\rvert}{\sum_{\mathbf{w}} \lvert\{\mathbf{w'} : \mathbf{C(w', w) > 0}\}\rvert} = \frac{\mathbf{1}}{\mathbf{50000}} = \mathbf{0.00002}$$

- *Conclusion:* Kneser-Ney correctly recognizes that *"Francisco"* has near-zero versatility as a general novel continuation, preventing catastrophic false predictions!

Topic 16: Hidden Markov Models (HMM) for Part-of-Speech Tagging

POS tagging maps a word sequence $W = (w_1, \ldots, w_T)$ to an optimal sequence of grammatical tags $T = (t_1, \ldots, t_T)$. In a generative HMM model:

$$\hat{\mathbf{T}} = \arg\max_{\mathbf{T}} \mathbf{P}(\mathbf{T} \mid \mathbf{W}) = \arg\max_{\mathbf{T}} \frac{\mathbf{P}(\mathbf{W} \mid \mathbf{T})\mathbf{P}(\mathbf{T})}{\mathbf{P}(\mathbf{W})} = \arg\max_{\mathbf{T}} \prod_{\mathbf{i=1}}^{\mathbf{T}} \underbrace{\mathbf{P}(\mathbf{w_i} \mid \mathbf{t_i})}_{\text{Emission } \mathbf{b_{t_i}(w_i)}} \underbrace{\mathbf{P}(\mathbf{t_i} \mid \mathbf{t_{i-1}})}_{\text{Transition } \mathbf{a_{t_{i-1},t_i}}}$$

The Viterbi Dynamic Programming Algorithm

Let $v_t(j)$ be the maximum probability of observing sequence $w_1 \dots w_t$ and ending in hidden state j at time t:

$$\mathbf{v_t(j)} = \max_{\mathbf{i=1}}^{\mathbf{|S|}} [\mathbf{v_{t-1}(i)} \cdot \mathbf{a_{ij}}] \cdot \mathbf{b_j(w_t)}$$

$$\text{Backpointer: } \mathbf{ptr_t(j)} = \arg\max_{\mathbf{i=1}}^{\mathbf{|S|}} [\mathbf{v_{t-1}(i)} \cdot \mathbf{a_{ij}}]$$

Solved Numerical 4: Step-by-Step Viterbi Trellis Trace for POS Tagging

Given 2 POS states $\{\text{Noun (N)}, \text{Verb (V)}\}$ and sentence $W = (\text{"Janet"}, \text{"will"}, \text{"back"})$.

Initial Probabilities: $\pi_N = 0.8, \pi_V = 0.2$.

Transitions (A): $P(N \mid N) = 0.4, P(V \mid N) = 0.6; \quad P(N \mid V) = 0.7, P(V \mid V) = 0.3$.

Emissions (B):

- $P(\text{"Janet"} \mid N) = 0.05, P(\text{"Janet"} \mid V) = 0.00$
- $P(\text{"will"} \mid N) = 0.01, P(\text{"will"} \mid V) = 0.04$
- $P(\text{"back"} \mid N) = 0.02, P(\text{"back"} \mid V) = 0.01$

Step 1: $t = 1$ ("Janet"):

- $v_1(N) = \pi_N \cdot b_N(\text{"Janet"}) = 0.8 \times 0.05 = \mathbf{0.0400}$ (ptr = Start)
- $v_1(V) = \pi_V \cdot b_V(\text{"Janet"}) = 0.2 \times 0.00 = \mathbf{0.0000}$ (ptr = Start)

Step 2: $t = 2$ ("will"):

- $v_2(N) = \max[v_1(N)a_{NN}, v_1(V)a_{VN}] \cdot b_N(\text{"will"}) = \max[0.0400 \times 0.4, 0] \times 0.01 = 0.0160 \times 0.01 = \mathbf{0.000160}$ ($\text{ptr}_2(N) = N$)
- $v_2(V) = \max[v_1(N)a_{NV}, v_1(V)a_{VV}] \cdot b_V(\text{"will"}) = \max[0.0400 \times 0.6, 0] \times 0.04 = 0.0240 \times 0.04 = \mathbf{0.000960}$ ($\text{ptr}_2(V) = N$)

Step 3: $t = 3$ ("back"):

- $v_3(N) = \max[v_2(N)a_{NN}, v_2(V)a_{VN}] \cdot b_N(\text{"back"}) = \max[0.000160 \times 0.4, 0.000960 \times 0.7] \times 0.02 = \max[0.000064, 0.000672] \times 0.02 = 0.000672 \times 0.02 = \mathbf{0.00001344}$ ($\text{ptr}_3(N) = V$)
- $v_3(V) = \max[v_2(N)a_{NV}, v_2(V)a_{VV}] \cdot b_V(\text{"back"}) = \max[0.000160 \times 0.6, 0.000960 \times 0.3] \times 0.01 = \max[0.000096, 0.000288] \times 0.01 = 0.000288 \times 0.01 = \mathbf{0.00000288}$ ($\text{ptr}_3(V) = V$)

Backtracking from Max Final State ($\max(v_3(N), v_3(V)) = v_3(N)$):

$$\hat{t}_3 = N \xrightarrow{\text{ptr}_3(N)} \hat{t}_2 = V \xrightarrow{\text{ptr}_2(V)} \hat{t}_1 = N \implies \hat{\mathbf{T}} = (\text{Noun, Verb, Noun})$$

Topic 17: Maximum Entropy Markov Models (MEMM) vs. Conditional Random Fields (CRF)

Model	Probability Formulation & Normalization	Key Architectural Trade-Off
HMM	Generative joint model: $P(W,T) = \prod P(t_i \mid t_{i-1})P(w_i \mid t_i)$.	Fast, but cannot incorporate overlapping, arbitrary non-independent lexical features.
MEMM	Discriminative: $P(T \mid W) = \prod P(t_i \mid t_{i-1}, \mathbf{x}_i) = \prod \frac{\exp(\mathbf{w}^T \mathbf{f}(t_i, t_{i-1}, \mathbf{x}))}{Z(t_{i-1}, \mathbf{x})}$. **Local per-state normalization**.	Suffers from the **Label Bias Problem**: States with few outgoing transitions dominate paths regardless of observation evidence!
Linear-Chain CRF	Discriminative: $P(T \mid W) = \frac{1}{Z(W)} \exp\left(\sum_{t=1}^{T} \sum_k w_k f_k(t_i, t_{i-1}, W, t)\right)$. **Global sequence-level normalization**.	Completely eliminates Label Bias; provably optimal discriminative sequence labeler.

Topic 18: Master University Examination Question Bank (Module 2)

Q1. Prove why Perplexity is equal to $2^{H(W)}$ for base-2 cross-entropy. (6 Marks)

Let empirical cross-entropy be $H(W) = -\frac{1}{N} \log_2 P(w_1, \ldots, w_N)$.
Then: $2^{H(W)} = 2^{-\frac{1}{N} \log_2 P(W)} = \left(2^{\log_2 P(W)}\right)^{-\frac{1}{N}} = (P(W))^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1, \ldots, w_N)}} \equiv \text{PP}(W)$. Q.E.D.

Q2. Explain the Label Bias Problem in MEMMs with a concrete state graph. (8 Marks)

In an MEMM, the transition probabilities from a given state must sum locally to 1 ($\sum_{s'} P(s' \mid s, x) = 1$). If state S_1 has only one outgoing transition to state S_2, then $P(S_2 \mid S_1, x) = 1.0$ regardless of how poorly the observation feature x matches state S_2! In contrast, a **Conditional Random Field (CRF)** normalizes globally across all possible entire sequences via partition function $Z(\mathbf{x}) = \sum_{y'} \exp(\sum_t \mathbf{w}^T \mathbf{f}(y_t, y_{t-1}, \mathbf{x}))$, allowing subsequent strong evidence to overturn locally constrained transitions!

Q3. Detail the Forward and Backward Algorithms in HMMs and their computational complexity. (8 Marks)

- **Forward Variable** $\alpha_t(j) = P(w_1 \ldots w_t, t_t = j \mid \lambda)$: Computes the total joint probability of observing the prefix sequence: $\alpha_t(j) = \sum_{i=1}^{|S|} \alpha_{t-1}(i) a_{ij} b_j(w_t)$.
- **Backward Variable** $\beta_t(i) = P(w_{t+1} \ldots w_T \mid t_t = i, \lambda)$: Computes the conditional probability of future suffix emissions: $\beta_t(i) = \sum_{j=1}^{|S|} a_{ij} b_j(w_{t+1}) \beta_{t+1}(j)$.

Both run in $O(T \cdot |S|^2)$ time, compared to naive exhaustive evaluation $O(|S|^T)$! Used together in the Baum-Welch (EM) algorithm to estimate expected transition and emission counts.

Topic 19: Advanced Smoothing Formalisms — Good-Turing & Witten-Bell

Good-Turing estimation reallocates probability mass based on the frequency of frequencies:

Good-Turing Frequency of Frequencies Formulation

Let N_r be the number of distinct n-grams that occurred exactly r times in the training corpus. The adjusted Good-Turing count r^* for an n-gram seen r times is:

$$\mathbf{r^* = (r+1)\frac{N_{r+1}}{N_r}}$$

$$\text{Total Probability Mass Assigned to Unseen Events } (\mathbf{r=0}): \ \mathbf{P_0 = \frac{N_1}{N}}$$

Where $N = \sum_{r=1}^{\infty} r N_r$ is the total number of observed token instances. The probability of an n-gram with count r is $P_{\text{GT}} = \frac{r^*}{N}$.

🏛️ Solved Numerical 7: Complete Good-Turing Discounting Calculation

In a corpus of $N = 100,000$ tokens, count statistics are: $N_1 = 20,000$ (singletons), $N_2 = 5,000$, $N_3 = 2,000$, $N_4 = 1,000$.

- **1. Probability mass reserved for all unseen words:**

$$P_0 = \frac{N_1}{N} = \frac{20,000}{100,000} = \mathbf{0.20} = \mathbf{20\%}$$

- **2. Adjusted count for singletons ($r = 1$):**

$$r_1^* = (1+1)\frac{N_2}{N_1} = 2 \times \frac{5,000}{20,000} = 2 \times 0.25 = \mathbf{0.50}$$

$$P_{\text{GT}}(r=1) = \frac{0.50}{100,000} = \mathbf{5.0 \times 10^{-6}}$$

- **3. Adjusted count for doubletons ($r = 2$):**

$$r_2^* = (2+1)\frac{N_3}{N_2} = 3 \times \frac{2,000}{5,000} = 3 \times 0.40 = \mathbf{1.20}$$

$$P_{\text{GT}}(r=2) = \frac{1.20}{100,000} = \mathbf{1.2 \times 10^{-5}}$$

Topic 20: HMM Training via the Baum-Welch (EM) Algorithm

When training sequences lack labeled hidden state annotations, the **Baum-Welch Algorithm** iteratively optimizes HMM parameters $\lambda = (A, B, \pi)$ to maximize marginal observation likelihood $P(W \mid \lambda)$:

1. **Expectation (E-step):** Compute forward probabilities $\alpha_t(i)$ and backward probabilities $\beta_t(i)$.

$$\gamma_t(\mathbf{i}) = \mathbf{P}(\mathbf{t_t = q_i} \mid \mathbf{W}, \lambda) = \frac{\alpha_t(\mathbf{i})\beta_t(\mathbf{i})}{\sum_{\mathbf{j}=1}^{|\mathbf{S}|} \alpha_t(\mathbf{j})\beta_t(\mathbf{j})}$$

$$\xi_t(\mathbf{i},\mathbf{j}) = \mathbf{P}(\mathbf{t_t = q_i, t_{t+1} = q_j} \mid \mathbf{W}, \lambda) = \frac{\alpha_t(\mathbf{i})\mathbf{a_{ij}}\mathbf{b_j}(\mathbf{w_{t+1}})\beta_{t+1}(\mathbf{j})}{\sum_{\mathbf{k}}\sum_{\mathbf{m}} \alpha_t(\mathbf{k})\mathbf{a_{km}}\mathbf{b_m}(\mathbf{w_{t+1}})\beta_{t+1}(\mathbf{m})}$$

2. **Maximization (M-step):** Re-estimate parameters by normalized expected event transitions:

$$\hat{\mathbf{a}}_{\mathbf{ij}} = \frac{\sum_{\mathbf{t}=1}^{\mathbf{T}-1} \xi_t(\mathbf{i},\mathbf{j})}{\sum_{\mathbf{t}=1}^{\mathbf{T}-1} \gamma_t(\mathbf{i})} \qquad \hat{\mathbf{b}}_{\mathbf{j}}(\mathbf{k}) = \frac{\sum_{\mathbf{t}=1,\mathbf{w_t}=\mathbf{v_k}}^{\mathbf{T}} \gamma_t(\mathbf{j})}{\sum_{\mathbf{t}=1}^{\mathbf{T}} \gamma_t(\mathbf{j})} \qquad \hat{\pi}_{\mathbf{i}} = \gamma_1(\mathbf{i})$$

Topic 24: Advanced Probability Formalisms & University Exam Problem Bank (Part II)

🏛️ Solved Numerical 10: Trigram Perplexity Calculation on Test Sentence

Given test sentence $S = \text{"~~I love NLP~~"}$ ($N = 4$ tokens: `I`, `love`, `NLP`, ``).

Learned Trigram Conditional Probabilities:

- $P(\text{"I"} \mid \text{"~~", "~~"}) = 0.25$
- $P(\text{"love"} \mid \text{"~~", "I"}) = 0.50$
- $P(\text{"NLP"} \mid \text{"I", "love"}) = 0.10$
- $P(\text{"~~"} \mid \text{"love", "NLP"}) = 0.20$

1. Joint Sentence Probability:

$$P(S) = 0.25 \times 0.50 \times 0.10 \times 0.20 = 0.0025 = 2.5 \times 10^{-3}$$

2. Test Set Perplexity ($N = 4$):

$$\text{PP}(\mathbf{S}) = \left(\frac{\mathbf{1}}{\mathbf{0.0025}} \right)^{\mathbf{1/4}} = (\mathbf{400})^{\mathbf{0.25}} \approx \mathbf{4.472}$$

Interpretation: At each word position, the language model is as uncertain as if choosing uniformly from among 4.47 possible words!

Q4. Compare Katz Backoff with Jelinek-Mercer Linear Interpolation. (8 Marks)

- **Jelinek-Mercer Interpolation:** Always combines higher and lower-order n-gram estimates ($P_{\text{JM}} = \lambda_3 P(w_i \mid w_{i-2}, w_{i-1}) + \lambda_2 P(w_i \mid w_{i-1}) + \lambda_1 P(w_i)$), even when high-order counts are large.
- **Katz Backoff:** Uses the high-order MLE count discounted by Good-Turing (d_r) if $C(w_{i-k}^{i}) > 0$. It backs off to lower-order models *only when* the high-order n-gram count is zero ($C = 0$), scaling the lower-order estimate by backoff normalization factor $\alpha(w_{i-k}^{i-1})$ so total probability sums strictly to 1.0!

Q5. Explain the Linear-Chain Conditional Random Field (CRF) Feature Functions and Viterbi Inference. (8 Marks)

In a linear-chain CRF, the conditional probability of label sequence $\mathbf{y}$ given observation sequence $\mathbf{x}$ is:

$$\mathbf{P}(\mathbf{y} \mid \mathbf{x}) = \frac{\mathbf{1}}{\mathbf{Z}(\mathbf{x})} \exp\left(\sum_{\mathbf{t}=\mathbf{1}}^{\mathbf{T}} \sum_{\mathbf{k}=\mathbf{1}}^{\mathbf{K}} \mathbf{w_k f_k}(\mathbf{y_t}, \mathbf{y_{t-1}}, \mathbf{x}, \mathbf{t}) \right)$$

Where f_k are arbitrary feature functions (e.g., $f_1 = 1$ if $y_t = \text{PROPN}$ and x_t starts with capital letter). Unlike HMMs, CRFs support arbitrary overlapping, non-independent features, and normalize globally via partition function $Z(\mathbf{x})$, completely eliminating the label bias problem!

Topic 24.3: Master University Exam Problem Bank (Part III)

🏛️ Solved Problem 11: Modified Kneser-Ney Smoothing with 3 Distinct Discount Parameters

Standard Kneser-Ney uses a single constant discount d. **Modified Kneser-Ney (Chen & Goodman 1996)** utilizes 3 distinct discounts d_1, d_2, d_{3+} based on whether the n-gram count is 1, 2, or ≥ 3:

$$\mathbf{d_1} = \mathbf{1} - \mathbf{2Y}\frac{\mathbf{N_2}}{\mathbf{N_1}}, \quad \mathbf{d_2} = \mathbf{2} - \mathbf{3Y}\frac{\mathbf{N_3}}{\mathbf{N_2}}, \quad \mathbf{d_{3+}} = \mathbf{3} - \mathbf{4Y}\frac{\mathbf{N_4}}{\mathbf{N_3}} \quad \text{where } \mathbf{Y} = \frac{\mathbf{N_1}}{\mathbf{N_1} + \mathbf{2N_2}}$$

Provides the most empirically accurate statistical language modeling probability estimates in computational linguistics!

Q6. Explain the Log-Probability Formulation in Viterbi Decoding to Prevent Numerical Underflow. (8 Marks)

In standard Viterbi decoding, sequence probabilities are computed via multiplying dozens of small floating-point emission and transition probabilities ($P \approx 10^{-4}$).

$$v_t(j) = \max_i [v_{t-1}(i) \cdot a_{ij}] \cdot b_j(w_t)$$

For a sentence of length $T = 30$, multiplying 30 small probabilities produces values below 10^{-60}, causing catastrophic 64-bit IEEE floating-point numerical underflow to zero (0.0).

- **Log-Viterbi Transformation:** Transform all operations into log-space:

$$\mathbf{V_t(j) = \max_i [V_{t-1}(i) + \ln(a_{ij})] + \ln(b_j(w_t))}$$

Replaces multiplications with additions and handles arbitrary sequence lengths T with zero precision loss!

Q7. Detail the Witten-Bell Smoothing Algorithm and Compare it with Good-Turing. (8 Marks)

Witten-Bell Smoothing models zero-count events as the arrival of a previously unseen token. It defines the probability of seeing a novel word based on the number of *distinct word types* $T(w_{i-1})$ that have already been observed following history w_{i-1}:

$$\mathbf{\lambda(w_{i-1}) = 1 - \frac{T(w_{i-1})}{T(w_{i-1}) + C(w_{i-1})} = \frac{C(w_{i-1})}{C(w_{i-1}) + T(w_{i-1})}}$$

$$P_{\text{WB}}(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1}) + T(w_{i-1})} + \frac{T(w_{i-1})}{C(w_{i-1}) + T(w_{i-1})} P(w_i)$$

Unlike Good-Turing (which requires global frequency-of-frequency counts N_r), Witten-Bell computes parameters directly from local history statistics.

🏛️ **Solved Problem 16: Linear-Chain CRF Forward-Backward Variable Recursion**

In a Linear-Chain CRF with potential functions $\psi_t(y_{t-1}, y_t, \mathbf{x}) = \exp(\sum_k w_k f_k(y_t, y_{t-1}, \mathbf{x}, t))$:

- **Forward Variable** $\alpha_t(j)$: Unnormalized score of prefix paths ending in state j at time t:

$$\mathbf{\alpha_t(j) = \sum_{i=1}^{|S|} \alpha_{t-1}(i) \psi_t(i, j, \mathbf{x}) \qquad \alpha_1(j) = \psi_1(\text{start}, j, \mathbf{x})}$$

- **Backward Variable** $\beta_t(i)$: Unnormalized score of suffix paths starting from state i at time t:

$$\mathbf{\beta_t(i) = \sum_{j=1}^{|S|} \psi_{t+1}(i, j, \mathbf{x}) \beta_{t+1}(j) \qquad \beta_T(i) = 1.0}$$

- **Global Partition Function:** $Z(\mathbf{x}) = \sum_{j=1}^{|S|} \alpha_T(j)$. Marginal probability: $P(y_t = j \mid \mathbf{x}) = \frac{\alpha_t(j)\beta_t(j)}{Z(\mathbf{x})}$.

Q8. Explain Cross-Entropy Loss in Language Model Training and its Relationship to Perplexity. (8 Marks)

During language model training with true one-hot next-word targets y_t and predicted probabilities $\hat{y}_t$, the token-level cross-entropy loss is: $$\mathcal{L}_t = -\sum_{w=1}^{|V|} y_{t,w} \log_2 \hat{y}_{t,w} = -\log_2 P(w_t \mid w_{Perplexity is the exponential of cross-entropy: $\text{PP}(W) = 2^{H(W)}$. Minimizing cross-entropy loss via gradient descent directly minimizes test set perplexity!

🏛 Solved Problem 17: Maximum Entropy / Logistic Regression POS Tagger Feature Weights

In a Maximum Entropy POS tagger, the conditional probability of tag t given context h is: $P(t \mid h) = \frac{1}{Z(h)} \exp(\sum_i \lambda_i f_i(h, t))$.

- Feature 1 (f_1): Word ends in `-ing` and $t = \text{VBG}$ ($\lambda_1 = +2.5$).
- Feature 2 (f_2): Previous tag is TO and $t = \text{VB}$ ($\lambda_2 = +3.1$).
- Feature 3 (f_3): Word starts with Capital and $t = \text{NNP}$ ($\lambda_3 = +4.0$).

🏛 Solved Problem 18: Trigram Jelinek–Mercer Expectation–Maximization λ Estimation

Linear interpolation weights $\lambda_1, \lambda_2, \lambda_3$ ($\sum \lambda_i = 1$) are learned on held-out validation corpus using EM:

$$\lambda_i^{(t+1)} = \frac{1}{N_{\text{heldout}}} \sum_{k=1}^{N_{\text{heldout}}} \frac{\lambda_i^{(t)} P_i(w_k \mid \text{history}_k)}{\sum_j \lambda_j^{(t)} P_j(w_k \mid \text{history}_k)}$$

🏛 Solved Problem 26: Cross-Entropy & Shannon Entropy in N-Gram Language Models

Prove that minimizing relative entropy (Kullback-Leibler Divergence $D_{\text{KL}}(P||Q)$) is strictly equivalent to minimizing cross-entropy loss $H(P, Q)$:

$$D_{\text{KL}}(P||Q) = \sum_x P(x) \log_2 \frac{P(x)}{Q(x)} = \sum_x P(x) \log_2 P(x) - \sum_x P(x) \log_2 Q(x) = -H(P) + H(P, Q)$$

$$\arg\min_{\mathbf{Q}} \mathbf{D}_{\text{KL}}(\mathbf{P}||\mathbf{Q}) \equiv \arg\min_{\mathbf{Q}} \mathbf{H}(\mathbf{P}, \mathbf{Q}) \quad \text{(Since true data entropy } \mathbf{H}(\mathbf{P}) \text{ is constant!)}$$

🏛 Solved Problem 27: Absolute Discounting with Interpolation Formula

Given discount $d = 0.75$: $P_{\text{Absolute}}(w_i \mid w_{i-1}) = \frac{\max(C(w_{i-1}, w_i) - 0.75, 0)}{C(w_{i-1})} + \frac{0.75 \times |\{w : C(w_{i-1}, w) > 0\}|}{C(w_{i-1})} P(w_i)$.

🏛 Solved Problem 38: Trigram Smoothing with Katz Backoff & Good–Turing Multipliers

In Katz Backoff, the probability of trigram $w_i \mid w_{i-2}, w_{i-1}$ is formulated as:

$$P_{\text{Katz}}(w_i \mid w_{i-2}, w_{i-1}) = \begin{cases} d_r \frac{C(w_{i-2}^i)}{C(w_{i-2}^{i-1})} & \text{if } C(w_{i-2}^i) > 0 \\ \alpha(w_{i-2}, w_{i-1}) P_{\text{Katz}}(w_i \mid w_{i-1}) & \text{if } C(w_{i-2}^i) = 0 \end{cases}$$

Where backoff weight $\alpha(w_{i-2}, w_{i-1}) = \frac{1 - \sum_{w:C>0} P_{\text{Katz}}(w|w_{i-2}, w_{i-1})}{1 - \sum_{w:C>0} P_{\text{Katz}}(w|w_{i-1})}$. Ensures exact conservation of total probability mass!

🏛 Solved Problem 46: Trigram Language Model Linear Interpolation Grid Search

Optimize weights $\lambda_1, \lambda_2, \lambda_3$ on validation cross-entropy $H(V) = -\frac{1}{N} \sum \log_2(\lambda_3 P_3 + \lambda_2 P_2 + \lambda_1 P_1)$:

- Trial 1: $(\lambda_1, \lambda_2, \lambda_3) = (0.1, 0.3, 0.6) \implies H(V) = 4.12 \text{ bits/token} \implies \text{PP} = 17.39$.
- Trial 2: $(\lambda_1, \lambda_2, \lambda_3) = (0.05, 0.25, 0.70) \implies H(V) = 3.98 \text{ bits/token} \implies \mathbf{\text{PP} = 15.78 \text{ (Optimal!)}}$.

🏛 Solved Problem 47: HMM Initial State & Transition Probability Smoothing

Apply Laplace smoothing to HMM transition matrix: $A_{ij} = \frac{C(t_{i-1}, t_i)+1}{\sum_k C(t_{i-1}, t_k)+|S|}$, guaranteeing non-zero transitions for all grammatical POS pairs!

🏛 Solved Problem 55: Forward-Backward State Posterior Decoding

In HMMs, state posterior probability $\gamma_t(i) = P(t_t = q_i \mid W) = \frac{\alpha_t(i)\beta_t(i)}{\sum_j \alpha_t(j)\beta_t(j)}$ allows computing the expected number of transitions from state i during Baum-Welch training!

🏛 Solved Problem 63: Trigram Good-Turing Discounting on Sparse Web Corpora

In a 1-billion-word web crawl, $N_1 = 40,000,000$ singleton trigrams, $N_2 = 10,000,000$ doubletons. $P_0 = \frac{40M}{1B} = \mathbf{0.040} = \mathbf{4.0\%}$ total probability mass is safely reserved for previously unseen novel trigram sequences!

🏛 Solved Problem 70: Viterbi Trellis Backpointer Array Proof

Prove by induction that the Viterbi dynamic programming backpointer array $\text{ptr}_t(j)$ recovers the globally optimal sequence $\hat{T}_{1..T}$ with zero heuristic approximation error!

🏛 Solved Problem 74: Trigram Perplexity vs Bigram Perplexity Comparative Analysis

On the Penn Treebank test corpus, standard Bigram MLE achieves Perplexity $\text{PP} \approx 170$, whereas Trigram with Kneser-Ney smoothing drops Perplexity to $\text{PP} \approx \mathbf{109}$, demonstrating a 35% reduction in predictive uncertainty!

🏛 Solved Problem 78: Maximum Entropy Markov Model (MEMM) Objective Function Proof

In MEMMs, parameters $\mathbf{w}$ are learned by maximizing conditional log-likelihood with L_2 Gaussian regularization: $\mathcal{L}(\mathbf{w}) = \sum_i \sum_t \left[\mathbf{w}^T \mathbf{f}(y_{i,t}, y_{i,t-1}, \mathbf{x}_i) - \ln Z(y_{i,t-1}, \mathbf{x}_i)\right] - \frac{\|\mathbf{w}\|^2}{2\sigma^2}$. Optimized via L-BFGS in linear time!

🏛 Solved Problem 85: Trigram Jelinek-Mercer vs Absolute Discounting

In Absolute Discounting: $P(w_i \mid w_{i-1}) = \frac{\max(C(w_{i-1}, w_i)-d, 0)}{C(w_{i-1})} + \frac{d \cdot |\{w : C(w_{i-1}, w) > 0\}|}{C(w_{i-1})} P(w_i)$. Subtracts constant $d \approx 0.75$ from all seen bigrams to fund lower-order unigram probability!

🏛 Solved Problem 91: Trigram Witten-Bell Dynamic Weight Trace

In Witten-Bell smoothing: $\lambda(w_{i-1}) = \frac{C(w_{i-1})}{C(w_{i-1})+T(w_{i-1})}$. If history w_{i-1} has count $C = 100$ and precedes $T = 20$ distinct word types, $\lambda = \frac{100}{120} = \mathbf{0.8333}$, reserving 16.67% mass for novel unseen transitions!

🏛 Solved Problem 97: HMM State Sequence Viterbi Dynamic Programming Invariance

Show that adding a constant $\log C$ to all transition and emission log-probabilities shifts all path scores uniformly without changing the $\arg\max$ argmax state sequence!

🏛️ Solved Problem 101: Absolute Discounting Modified Interpolation Proof

Prove that for any vocabulary V, the sum of smoothed conditional probabilities $\sum_{w \in V} P_{\text{Absolute}}(w \mid w_{i-1}) = 1.0$ strictly conserves probability mass.