ABHYASMITRA.IN

COMPREHENSIVE UNIVERSITY STUDY NOTES

NATURAL LANGUAGE PROCESSING

Unit III — Statistical NLP Models and Language Modeling

Topics Covered in this Unit

Probability Foundations in NLP: Conditional Probability, Chain Rule & Maximum Likelihood Estimation (MLE)
N-Gram Language Models: Unigram, Bigram, Trigram Formulations & The Markov Independence Assumption
Model Evaluation: Perplexity (PP) Mathematical Derivation & Cross-Entropy Relationships
Smoothing, Backoff & Interpolation: Laplace (Add-1), Good-Turing, Katz Backoff & Jelinek-Mercer / Kneser-Ney
Statistical Text Classification: Multinomial Naive Bayes (Laplace Smoothed) & Logistic Regression (MaxEnt)
Tagging Models: Hidden Markov Models (HMM 5-Tuple: States, Observations, Transition A, Emission B Matrices)
The Viterbi Decoding Algorithm: Dynamic Programming Trellis, Forward Equations & Backpointers
Probabilistic Parsing: Probabilistic Context-Free Grammars (PCFG), Rule Probabilities & Parse Disambiguation
Parsing Algorithms: Top-Down vs. Bottom-Up Parsing & The CKY (Cocke-Kasami-Younger) Algorithm in CNF
High-Yield University Exam Question Bank with Full Numerical Derivations & Unit Summary

Compiled in a structured, easy-to-understand, textbook style format
Specially curated for B.Tech / B.E. / BCA / MCA / B.Sc Computer Science & Data Science Students

COURSE SYLLABUS & MAPPING

[UNIT III SYLLABUS — PRESCRIBED UNIVERSITY CURRICULUM]

This section contains the official syllabus for Unit III. Verify with your university curriculum mapping.

Unit III Syllabus Breakdown:

Unit III Statistical NLP Models and Language Modeling: Probability Basics: Conditional Probability concept and applications in NLP; Maximum Likelihood Estimation (MLE)—parameter estimation techniques for languages and statistical models. N-gram Language Models: Unigram, bigram, trigram, Training, perplexity, Backoff and interpolation. Classification Models: Naive Bayes for text, Logistic Regression, Evaluation for text classification. Tagging Models: Hidden Markov Models (HMMs), Viterbi algorithm, HMM-based POS tagging. Parsing with Probabilities: PCFG, Probabilistic parsing, Top-down and bottom-up parsing.

• Course Code: _____ • Semester / Year: _____

TABLE OF CONTENTS

3.1 Probability Foundations & Maximum Likelihood Estimation

Probability Basics in Statistical NLP

• *Conditional Probability: P(A | B) = P(A intersect B) / P(B)*

• *Chain Rule of Probability: For a sequence of words W = (w_1, w_2, ..., w_n):*

*P(w_1, w_2, ..., w_n) = P(w_1) * P(w_2 | w_1) * P(w_3 | w_1, w_2) * ... * P(w_n | w_1, ..., w_{n-1}) = prod_{i=1}^n P(w_i | w_1, ..., w_{i-1})*

• *Maximum Likelihood Estimation (MLE): Parameter estimation technique that selects parameter values theta maximizing the likelihood function L(theta) = P(Data | theta).*

3.2 N-Gram Language Models & Perplexity

1. The Markov Assumption & N-Gram Formulations

Computing joint sentence probabilities using the full chain rule is impossible due to infinite historical conditioning. The Markov Assumption approximates history by assuming the probability of an upcoming word depends only on the preceding (N - 1) words:

- **1. Unigram Model (N=1):** Assumes all words occur completely independently: P(w_1, ..., w_n) = prod_{i=1}^n P(w_i) = prod (count(w_i) / N_total).
- **2. Bigram Model (N=2):** Conditioned on the immediately preceding single word: P(w_i | w_1, ..., w_{i-1}) approx P(w_i | w_{i-1}). Bigram MLE Formula: P_MLE(w_i | w_{i-1}) = count(w_{i-1}, w_i) / count(w_{i-1}).
- **3. Trigram Model (N=3):** Conditioned on preceding two words: P(w_i | w_{i-2}, w_{i-1}) = count(w_{i-2}, w_{i-1}, w_i) / count(w_{i-2}, w_{i-1}).

2. Model Evaluation: Perplexity (PP)

Mathematical Formulation: Perplexity

Perplexity (PP) is the standard quantitative metric for evaluating language models on an unseen test set W = (w_1, w_2, ..., w_N):

*PP(W) = P(w_1, w_2, ..., w_N)^(-1 / N) = sqrt[N]{ 1 / prod_{i=1}^N P(w_i | w_{i-1}) } = 2^(- (1/N) * sum_{i=1}^N log_2 P(w_i | w_{i-1}))*

• *Intuition: Perplexity represents the effective Branching Factor (the weighted average number of equally likely words the model is choosing among at each step).*

• *Key Principle: LOWER perplexity indicates a SUPERIOR language model with higher predictive probability on test text.*

3. Smoothing, Backoff & Interpolation

If an unseen bigram appears during testing, count(w_{i-1}, w_i) = 0, causing P(W) = 0 and Perplexity to explode to infinity (Zero-Probability Problem). Smoothing reallocates probability mass from frequent words to unseen zero-count events.

- **1. Laplace (Add-1) Smoothing:** P_Laplace(w_i | w_{i-1}) = [count(w_{i-1}, w_i) + 1] / [count(w_{i-1}) + |V|] (where |V| is vocabulary size).
- **2. Add-k (Lidstone) Smoothing:** P_Lidstone(w_i | w_{i-1}) = [count(w_{i-1}, w_i) + k] / [count(w_{i-1}) + k * |V|] (where 0 < k < 1, e.g., k = 0.05).
- **3. Jelinek-Mercer Linear Interpolation:** Combines trigram, bigram, and unigram probabilities linearly: P_interp(w_i | w_{i-2}, w_{i-1}) = lambda_1 * P(w_i | w_{i-2}, w_{i-1}) + lambda_2 * P(w_i | w_{i-1}) + lambda_3 * P(w_i) (where sum lambda_i = 1).
- **4. Katz Backoff:** Uses higher-order trigram if count > 0; otherwise backs off to discounted bigram / unigram scaled by backoff weight alpha.
- **5. Kneser-Ney Smoothing:** State-of-the-art smoothing using Continuation Probability (how likely a word is to complete an unseen context, e.g., 'Francisco' is frequent only after 'San', so unigram continuation probability is low).

3.3 Statistical Text Classification Models

1. Multinomial Naive Bayes Classifier

Multinomial Naive Bayes applies Bayes' Theorem with the Naive Conditional Independence Assumption: all feature word tokens occur independently given the document class c.

- **Classification Decision Rule:** c_hat = argmax_{c in C} [log P(c) + sum_{i=1}^n log P(w_i | c)].
- **Laplace Smoothed Parameters:**
  - Class Prior: P(c) = N_c / N_total (fraction of documents in class c).
  - Word Likelihood: P(w_i | c) = [count(w_i, c) + 1] / [sum_{w in V} count(w, c) + |V|].

2. Logistic Regression (Maximum Entropy / MaxEnt Classifier)

Logistic Regression directly models the conditional posterior class distribution P(c | d) via a log-linear model: P(c | d) = exp(w_c^T x + b_c) / sum_{c' in C} exp(w_{c'}^T x + b_{c'}).

- **Loss Function:** Trained by minimizing Cross-Entropy Loss (Negative Log-Likelihood) via Gradient Descent: L = - sum_{i=1}^N log P(y_i | x_i).

3.4 Tagging Models: Hidden Markov Models & Viterbi

Formal Definition: Hidden Markov Model (HMM 5-Tuple)

A Hidden Markov Model is a generative statistical sequential model defined as a 5-tuple: lambda = (S, V, A, B, pi)

- *S = {s_1, s_2, ..., s_N}: Finite set of Hidden States (e.g., POS Tags: NN, VB, JJ, DT).*
- *V = {v_1, v_2, ..., v_M}: Finite set of Observable Symbols (e.g., Words: 'the', 'cat', 'runs').*
- *A = {a_{ij}}: State Transition Probability Matrix: a_{ij} = P(t_{k} = s_j | t_{k-1} = s_i).*
- *B = {b_{j}(k)}: Emission / Output Probability Matrix: b_{j}(k) = P(w_k | t_j = s_j).*
- *pi = {pi_i}: Initial State Probability Distribution: pi_i = P(t_1 = s_i).*

HMM POS Tagging Formulation

Given observable word sequence W = (w_1, w_2, ..., w_T), find the most probable hidden POS tag sequence T* = (t_1, t_2, ..., t_T):

- **Decoding Objective:** T* = argmax_{t_1..t_T} P(w_1..w_T, t_1..t_T) approx argmax prod_{i=1}^T [P(t_i | t_{i-1}) * P(w_i | t_i)].

The Viterbi Decoding Algorithm (Dynamic Programming)

The Viterbi algorithm computes the most probable state sequence in O(T * N^2) time using a dynamic programming trellis v_t(j) representing the probability of the most likely path ending in state j at step t:

- **1. Initialization:** v_1(j) = pi_j * b_j(w_1); backpointer_1(j) = 0 (for 1 <= j <= N).
- **2. Recursion (for t = 2 to T, for j = 1 to N):**
  - v_t(j) = max_{i=1..N} [v_{t-1}(i) * a_{ij}] * b_j(w_t)
  - backpointer_t(j) = argmax_{i=1..N} [v_{t-1}(i) * a_{ij}]
- **3. Termination:** P* = max_{i=1..N} v_T(i); t_T* = argmax_{i=1..N} v_T(i).
- **4. Backtracking:** For t = T-1 down to 1: t_t* = backpointer_{t+1}(t_{t+1}*).

3.5 Probabilistic Context-Free Grammars (PCFG)

Definition: Probabilistic Context-Free Grammar (PCFG)
A PCFG augments a standard Context-Free Grammar G = (V, Sigma, R, S) by assigning a probability P(A -> alpha) to each production rule such that for every non-terminal A in V: sum_{alpha} P(A -> alpha) = 1.0.

Probability of a Parse Tree & Disambiguation

- **Parse Tree Probability:** The probability of a complete parse tree T is the product of the probabilities of all production rules r in T used to derive the sentence: P(T, S) = prod_{r in T} P(r).
- **Syntactic Disambiguation:** If a sentence S has multiple valid parse trees {T_1, T_2}, the most probable parse tree T* is selected: T* = argmax_{T} P(T, S).

The CKY (Cocke-Kasami-Younger) Parsing Algorithm

The CKY algorithm is a dynamic programming algorithm that parses sentences in Chomsky Normal Form (CNF where all rules are A -> B C or A -> w) in O(n^3 * |R|) time by building a triangular chart table storing non-terminal spans.

3.6 High-Yield University Exam Question Bank

Part A: Short Answer Questions (2 to 5 Marks)

Q1. What is Perplexity in language modeling? What does lower perplexity signify? [3 Marks]

- **Answer:** Perplexity is the inverse geometric mean probability of an unseen test sequence: PP(W) = P(w_1..w_N)^(-1/N). It represents the effective branching factor (number of words the model is confused between). Lower perplexity indicates superior predictive accuracy.

Q2. Define the 5 components of a Hidden Markov Model (HMM). [3 Marks]

- **Answer:** HMM lambda = (S, V, A, B, pi): S = Hidden state set (POS tags), V = Observation vocabulary (words), A = State transition probability matrix P(t_i|t_{i-1}), B = Emission probability matrix P(w_i|t_i), and pi = Initial state probability vector.

Q3. State the formula for Laplace (Add-1) smoothing in bigram language models. [3 Marks]

- **Answer:** P_Laplace(w_i | w_{i-1}) = [count(w_{i-1}, w_i) + 1] / [count(w_{i-1}) + |V|], where |V| is the total vocabulary size.

Part B: Long Answer Questions (10 to 15 Marks Outline Guides)

- **Q4. Explain N-Gram Language Models (Unigram, Bigram, Trigram) with MLE parameter derivations and Markov assumptions. [12 Marks]**
- **Q5. Explain Smoothing techniques (Laplace, Lidstone, Jelinek-Mercer Interpolation, Katz Backoff, Kneser-Ney) to solve the zero-probability problem. [12 Marks]**
- **Q6. Formulate the Viterbi Algorithm step-by-step with initialization, recursion, termination, and backtracking formulas. [12 Marks]**
- **Q7. Explain Probabilistic Context-Free Grammars (PCFG) and the CKY dynamic programming parsing algorithm in Chomsky Normal Form (CNF). [10 Marks]**

3.7 Unit Summary & Key Takeaways

- Statistical NLP models sentence probabilities via Markov N-gram approximations: Unigram, Bigram P(w_i|w_{i-1}), and Trigram.
- Perplexity PP(W) = P(W)^(-1/N) measures language model quality (lower is better).
- Smoothing methods (Laplace, Interpolation, Kneser-Ney) prevent zero-probability crashes.
- Multinomial Naive Bayes uses class priors and smoothed word likelihoods for text classification.
- HMM sequence tagging relies on transition matrix A and emission matrix B, decoded optimally via the Viterbi dynamic programming algorithm.
- PCFGs assign rule probabilities P(A -> alpha) to compute parse tree likelihoods P(T) = prod P(r_i) and resolve ambiguity.