NLP Exam Questions & Answers: Units 4 & 5

UNIT 4

Question 1: Explain HMM (Hidden Markov Model) for Part-of-Speech Tagging with an example

Answer:

Hidden Markov Models (HMMs) are probabilistic sequence models used for Part-of-Speech (POS) tagging. HMMs have two main components: states (POS tags) and observations (words), with transition probabilities between tags and emission probabilities for words given tags.

Key Components:

- **States**: POS tags (Noun, Verb, Modal, etc.)
- **Observations**: Words in sentences
- **Transition Probabilities**: P(tag_i | tag_i-1)
- **Emission Probabilities**: P(word | tag)

How It Works:

1. **Emission Probabilities**: How likely a word belongs to a particular POS tag
    - P(Mary | Noun), P(will | Modal), P(see | Verb)
2. **Transition Probabilities**: How likely one POS tag follows another
    - P(Modal | Noun), P(Verb | Modal), P(Noun | Verb)

Example with Training Corpus:

Training data:

- "Martin Justin can watch Will" → N N M V N
- "Spot will watch Martin" → N M V N
- "Justin spot Martin" → N V N
- "Martin will pat Spot" → N M V N

Emission Probabilities (from training):

- P(Martin | Noun) = 4/9
- P(Will | Noun) = 1/9, P(Will | Modal) = 3/4
- P(spot | Noun) = 2/9, P(spot | Verb) = 1/5

Transition Probabilities:

- P(N | S) = 3/4 (Noun after Start)
- P(M | N) = 1/3 (Modal after Noun)
- P(V | M) = 3/4 (Verb after Modal)
- P(N | V) = 1 (Noun after Verb)

Test Sentence: "Justin will spot Will"

For each word combination, calculate: P(word | tag) × P(tag | previous_tag)

- "Justin": P(Justin | N) × P(N | S) = (2/9) × (3/4) = 2/12
- "will" as Modal: P(will | M) × P(M | N) = (3/4) × (1/3) = 1/4
- "spot" as Verb: P(spot | V) × P(V | M) = (1/5) × (3/4) = 3/20

Result: Justin (N) will (M) spot (V) Will (N)

HMM successfully disambiguates words based on context using learned probabilities.

Question 2: Explain Conditional Random Fields (CRF) for Named Entity Recognition

Answer:

CRF (Conditional Random Fields) is a discriminative probabilistic model that directly models conditional probability P(Y|X), unlike HMMs which model joint probability. CRFs are more effective for Named Entity Recognition (NER).

Named Entity Recognition (NER) Categories:

- **PER**: People (Ramu, Seeta)
- **ORG**: Organizations (Google, Facebook)
- **LOC**: Locations (Mt. Sanitas, India)
- **GPE**: Geo-Political Entities (Palo Alto, USA)

Key Challenges in NER:

1. **Segmentation Ambiguity**: "New York" - single entity or two words?
2. **Tag Assignment Ambiguity**: "Nirma" - person name (PER) or brand (ORG)?

How CRF Works:

Feature Functions: Return 1 (True) or 0 (False) based on conditions

- f_1: Is the word capitalized?
- f_2: Does it contain numbers?
- f_3: Is it preceded by "Mr." or "Dr."?
- f_4: Does it appear in a gazetteer?

Example: Sentence "Ram is cool" with labels PER O O

CRF Probability Formula:

$P(Y|X) = (1/Z) \times \exp(\Sigma\ \Sigma\ w_k F_k(X, Y))$

Where:

- Y = label sequence
- X = word sequence
- w_k = weights for feature functions
- Z = normalization constant

Advantages:

- Uses arbitrary feature functions
- Models label dependencies
- More flexible than HMMs
- Doesn't assume independence

Evaluation Metrics:

- **Precision**: Correctly labeled / Total labeled
- **Recall**: Correctly labeled / Total should be labeled
- **F1-Score**: Harmonic mean of precision and recall

Question 3: Explain the NER (Named Entity Recognition) process in detail

Answer:

NER is the task of identifying and classifying named entities in text into predefined categories. It's crucial for Information Extraction.

NER Process Steps:

1. Data Preparation

- **Sentence Boundary Segmentation**: Split text into sentences
- **Word Tokenization**: Break sentences into words
- **POS Tagging**: Assign part-of-speech tags

2. Named Entity Identification
Find entities by:

- Searching tokens against knowledge base (KB)
- Using pattern matching and linguistic rules
- Applying statistical models

3. Named Entity Classification
Assign identified entities to categories using:

- Statistical models (CRF, HMM)
- Machine learning classifiers
- Deep learning models (LSTM, BERT)

4. Statistical Testing
Apply statistical tests to select best classification when multiple results exist.

Example:

Input Text: "Citing high fuel prices, United Airlines said Friday it has increased fares by $6 per round trip on flights to some cities also served by lower-cost carriers. American Airlines, a unit of AMR Corp., immediately matched the move, spokesman Tim Wagner said."

NER Output:

- **United Airlines** → ORG
- **Friday** → TIME
- **$6** → MONEY
- **American Airlines** → ORG
- **AMR Corp.** → ORG
- **Tim Wagner** → PER

Key Challenges:

1. **Boundary Detection**: Is "New York" one entity or two?
2. **Context Dependency**: "Washington" - person, city, or university?
3. **Multiple Meanings**: "Apple" - fruit or company?
4. **Handling abbreviations and acronyms**
5. **Nested entities**: "Bank of America Stadium"

Approaches to NER:

1. Rule-Based

- Hand-crafted patterns and gazetteers
- Regular expressions
- High precision, low coverage

2. Machine Learning

- CRF (most popular)
- HMM, Maximum Entropy
- Requires feature engineering

3. Deep Learning

- BiLSTM-CRF models
- BERT-based models
- Transformer architectures
- Automatic feature learning

Evaluation Metrics:

- **Precision**: Correct entities / Total entities identified
- **Recall**: Correct entities / Total entities in text
- **F1-Score**: 2 × (Precision × Recall) / (Precision + Recall)

UNIT 5

Question 1: Explain Language Divergence in Machine Translation

Answer:

Language divergence refers to systematic differences between languages that must be addressed in Machine Translation (MT) systems. While some language aspects are universal, languages differ significantly in structure, vocabulary, and expression.

Types of Language Divergences:

1. Word Order Typology

Languages vary in subject (S), verb (V), and object (O) ordering:

- **SVO Languages**: English, French, German, Mandarin
  - "He wrote a letter to a friend"
- **SOV Languages**: Hindi, Japanese
  - Japanese: "tomodachi ni tegami-o kaita" (friend to letter wrote)
- **VSO Languages**: Irish, Arabic
  - Arabic: "katabt risala li sadiq" (wrote letter to friend)

Impact on MT: Systems must reorder words appropriately.

2. Lexical Divergences

Languages divide conceptual space differently:

- **One-to-Many**: German "Wand" (indoor) vs "Mauer" (outdoor wall)
- **Many-to-Many**: Overlapping but non-identical word meanings
- **Lexical Gaps**: Missing direct equivalents
  - Mandarin "孝" (xiào) - filial piety
  - Japanese "親孝行" (oyakokō) - devotion to parents

3. Morphological Typology

- **Isolating Languages**: Vietnamese (one morpheme per word)
- **Agglutinative Languages**: Turkish, Finnish (multiple clear morphemes)
- **Polysynthetic Languages**: Inuktitut (entire sentences in single words)

Impact: Systems must handle subword-level structure.

4. Referential Density

- **Pro-drop Languages**: Spanish, Japanese (can omit pronouns)
  - "Habla español" (speaks Spanish)
- **Non-pro-drop**: English (require pronouns)
  - "She speaks Spanish"

Implications for MT:

- Architecture must handle varying structures
- Training data must cover diverse language pairs
- Subword models essential for morphologically rich languages
- Context modeling critical for implicit information

Question 2: Explain Encoder-Decoder Model with RNN architecture

Answer:

The Encoder-Decoder architecture is fundamental for sequence-to-sequence (seq2seq) tasks like machine translation, summarization, and question answering.

Three Main Components:

1. Encoder

- Processes input sequence (source sentence)
- Uses RNN (LSTM or GRU)
- Compresses input into fixed-size context vector
- Updates hidden state at each time step

2. Context Vector

- Fixed-size representation of entire input
- Final hidden state of encoder
- Contains semantic meaning of source
- Passed to decoder as initial state

3. Decoder

- Generates output sequence (target sentence)
- Another RNN taking context vector as initial state
- Produces one token at a time
- Continues until end-of-sequence token

Working Mechanism:

Encoding Phase:

Input "I love NLP" → [I, love, NLP]

1. Encoder reads: "I" → h_1
2. Encoder reads: "love" → h_2
3. Encoder reads: "NLP" → h_3
4. Context vector $c = h_3$

Mathematical Representation:
h_t = f(h_{t-1}, x_t)
c = h_n (context vector)

Decoding Phase:

1. Decoder initialized with c
2. At each time step:
    - Takes previous hidden state
    - Takes previously generated token
    - Generates next token
3. Continues until <END> token

Mathematical Representation:
$s_t = g(s_{t-1}, y_{t-1}, c)$
$y_t = \text{softmax}(W_s s_t)$

Example: English to French

Input: "I love NLP"
Target: "J'aime le PNL"

Encoding:

- h_3 = context vector (compressed)

Decoding:

- $s_0 = c$
- Generate y_1 = "J'" based on s_0
- Generate y_2 = "aime" based on s_1
- Generate y_3 = "le" based on s_2
- Generate y_4 = "PNL" based on s_3
- Generate y_5 = <END>

Key Features:

Advantages:

1. Variable length handling
2. End-to-end training
3. Contextual understanding
4. Flexibility with different lengths

Limitations:

1. **Information Bottleneck**: Entire input compressed to single vector
2. **Long-Range Dependencies**: RNNs struggle with very long sequences
3. **No Alignment**: Decoder doesn't know relevant input parts

Training Process:

1. **Teacher Forcing**: Use actual target tokens during training
2. **Loss Function**: Cross-entropy between predicted and actual
3. **Backpropagation**: Through both encoder and decoder
4. **Optimization**: Minimize loss

Question 3: Explain Attention Mechanism in Neural Machine Translation

Answer:

The attention mechanism allows neural networks to focus selectively on relevant parts of input when generating output. It addresses the information bottleneck of basic encoder-decoder models.

Problem with Basic Encoder-Decoder:

- Entire input compressed into single context vector
- Information loss for long sequences
- Cannot focus on specific relevant input parts
- Poor performance on longer sentences

How Attention Works:

Step 1: Calculate Alignment Scores

$\text{score}(s_t, h_i) = s_t^T W_a h_i$

Where:

- s_t = current decoder hidden state
- h_i = i-th encoder hidden state
- W_a = learnable weight matrix

Step 2: Compute Attention Weights

$\alpha_{t,i} = \exp(\text{score}(s_t, h_i)) / \Sigma \exp(\text{score}(s_t, h_j))$

Properties:

- $\alpha_{t,i} \in [0, 1]$
- $\Sigma \alpha_{t,i} = 1$

Step 3: Generate Context Vector

$c_t = \Sigma \alpha_{t,i} \cdot h_i$

Emphasizes relevant input parts for current output.

Step 4: Produce Output Token

$s_t = g(s_{t-1}, y_{t-1}, c_t)$
$y_t = \text{softmax}(W_s[s_t; c_t])$

Example: English to French Translation

Input: "I love NLP"
Target: "J'aime le PNL"

Generating "J'aime":

- Alignment Scores: $\text{score}(s_1, h_1)=0.8$, $\text{score}(s_1, h_2)=0.6$, $\text{score}(s_1, h_3)=0.1$
- Attention Weights: $\alpha_{1,1}=0.5$, $\alpha_{1,2}=0.4$, $\alpha_{1,3}=0.1$
- Context: $c_1 = 0.5 \times h_1 + 0.4 \times h_2 + 0.1 \times h_3$
- Output: "J'aime"

Generating "le":

- Alignment shifts toward "NLP"
- Attention Weights: $\alpha_{2,1}=0.15$, $\alpha_{2,2}=0.25$, $\alpha_{2,3}=0.60$
- Context: $c_2 = 0.15 \times h_1 + 0.25 \times h_2 + 0.60 \times h_3$
- Output: "le"

Generating "PNL":

- Attention heavily focuses on h_3
- $\alpha_{3,3} \approx 0.9$
- Output: "PNL"

Key Advantages:

1. **No Information Bottleneck**
    - Each decoder step accesses all encoder states
    - Can selectively focus on relevant information
    - Long sequences handled better
2. **Interpretability**
    - Attention weights show which input words influenced output
    - Provides alignment visualization
    - Helps understand model decisions
3. **Better Performance**
    - Significantly improves translation quality
    - Effective for long sentences
    - Handles complex alignments
4. **Flexible Architecture**
    - Works with any encoder-decoder model
    - Compatible with RNN, LSTM, GRU, Transformers
    - Foundation for modern NLP architectures

Types of Attention:

1. **Global Attention (Soft)**: Considers all source states
2. **Local Attention**: Focuses on subset of positions
3. **Self-Attention**: Used in Transformers

The attention mechanism revolutionized seq2seq models and became foundation for Transformer architectures.

Question 4: Explain Beam Search algorithm with example

Answer:

Beam Search is a heuristic search algorithm used for decoding sequences in NLP tasks. It improves upon greedy search by considering multiple candidate sequences simultaneously.

Motivation:

Greedy Search Problem:

- Selects highest probability token at each step
- Makes locally optimal choices
- May miss globally optimal sequence
- Cannot recover from early mistakes

Beam Search Solution:

- Maintains top-B candidates at each step
- Explores multiple paths in parallel
- Finds better overall sequences
- Balances exploration vs. computational cost

Key Concepts:

1. Beam Width (B)

- Number of candidate sequences retained
- B=1 reduces to greedy search
- Larger B → more accurate but slower
- Typical values: B=5 to B=10

2. Beam Score

- Cumulative log-probability of sequence
- Measures sequence likelihood

score(y) = log P(y|x) = Σ log P(y_i | y_1, ..., y_{i-1}, x)

Why log-probabilities?

- Prevents numerical underflow
- Converts products to sums

Beam Search Algorithm:

Step 1: Initialize

- Start with empty sequence
- Generate first token candidates
- Select top-B based on probability

Step 2: Expand
For each of B candidates:

- Generate all possible next tokens
- Compute cumulative score for each extension
- Results in B × V candidates

Step 3: Select

- Keep only top-B candidates overall
- Prune lower-scoring sequences

Step 4: Repeat

- Continue until all beams generate <END>
- Or reach maximum length

Step 5: Final Selection

- Choose sequence with highest score
- May apply length normalization

Detailed Example: English to French

Task: Translate "I love NLP"
Beam Width: B = 2

Step 1: First Word

Initial candidates:

- "J'" → score = -0.5
- "Le" → score = -1.0

Top 2 beams: ["J'", "Le"]

Step 2: Second Word

Expanding "J'":

- "J' aime" → score = -0.8
- "J' le" → score = -1.7

Expanding "Le":

- "Le aime" → score = -1.7
- "Le le" → score = -2.1

Select top 2:

- "J' aime" → -0.8 ✓
- "J' le" → -1.7 ✓

Step 3: Third Word

Expanding "J' aime":

- "J' aime le" → -1.0 ✓
- "J' aime NLP" → -1.4

Expanding "J' le":

- "J' le aime" → -2.2
- "J' le le" → -2.6

Select top 2:

- "J' aime le" → -1.0
- "J' aime NLP" → -1.4

Continue until <END>

Final Output: "J' aime le" (score = -1.0)

Length Normalization:

score(y) = (1/|y|) × Σ log P(y_i | y_1, ..., y_{i-1}, x)

Or with penalty:

score(y) = (1/|y|^α) × Σ log P(y_i | y_1, ..., y_{i-1}, x)

Advantages:

1. Better quality than greedy
2. Explores multiple options
3. Computationally tractable
4. Tunable beam width

Disadvantages:

1. Not globally optimal
2. Exposure bias between training and testing
3. Limited diversity among beams
4. B times slower than greedy

Question 5: Explain MT Evaluation using BLEU score

Answer:

BLEU (Bilingual Evaluation Understudy) is the most widely used automatic metric for MT evaluation. It measures similarity between machine translation and human reference translations using n-gram overlap.

Key Idea: Good translations have high n-gram overlap with reference translations.

BLEU Score Components:

1. Precision

Basic precision counts matching words:

Precision = (Words matching reference) / (Total words in output)

Example:

- Output: "She drinks the milk"
- Reference: "She drank the milk"
- Matches: "She", "the", "milk" = 3/4 = 0.75

Problems:

1. **Repetition**: Rewards repeated words
2. **Multiple References**: Doesn't utilize alternatives

2. Clipped Precision

Bounds word count by maximum occurrence in any reference:

Example:

- Output: "She She She eats a sour cherry"
- Reference 1: "She is eating a blueberry as she loves it"
- "She" appears 3 times in output, max 2 in refs → clipped = 2

Clipped Precision = 4/7 = 0.57

3. N-gram Precision

BLEU uses multiple n-gram lengths:

Example:

- Output: "the cat is on the mat"
- Reference: "the cat sat on the mat"

Unigrams: 5/6
Bigrams: 3/5
Trigrams: 1/4

4. BLEU Score Calculation

BLEU combines multiple n-gram precisions using geometric mean:

$BLEU_N = BP \times \exp(\Sigma w_n \log p_n)$

Where:

- N = maximum n-gram length (typically 4)
- w_n = weight for n-gram precision
- BP = brevity penalty

BLEU-4 Formula:
$BLEU\text{-}4 = BP \times (p_1 \times p_2 \times p_3 \times p_4)^{1/4}$

5. Brevity Penalty (BP)

Prevents artificially high scores for short outputs.

Problem: Output "the" matches reference perfectly (1.0 precision)

Brevity Penalty Formula:

BP = 1 if $c > r$
BP = $e^{(1-r/c)}$ if $c \le r$

Where:

- c = length of candidate (output)
- r = length of reference

Example:

- c = 5 words, r = 6 words
- BP = $e^{(1-6/5)} = e^{(-0.2)} \approx 0.82$

Complete BLEU Example:

Output: "the cat the mat"
Reference: "the cat is on the mat"

Step 1: Calculate precisions

- Unigrams: 4/4 = 1.0
- Bigrams: 2/3 = 0.67
- Trigrams: 0/2 = 0
- 4-grams: 0/1 = 0

Step 2: Geometric mean
GM = $(1.0 \times 0.67 \times 0.01 \times 0.01)^{(1/4)} \approx 0.14$

Step 3: Brevity penalty

- $c = 4$, $r = 6$
- BP = $e^{(1-6/4)} = 0.61$

Step 4: Final BLEU
BLEU = $0.61 \times 0.14 \approx 0.085$ (very low)

BLEU Score Interpretation:

Score	Quality
< 0.10	Almost useless
0.10-0.20	Poor
0.20-0.30	Understandable
0.30-0.40	Good
0.40-0.50	High quality
0.50-0.60	Very high
> 0.60	Often better than human

Advantages:

1. Fast and automatic

2. Reproducible
3. Language-independent
4. Correlates with human judgment
5. Widely adopted

Limitations:

1. Recall not considered
2. Synonyms not recognized
3. Doesn't handle paraphrases well
4. Requires multiple references
5. Not semantic (focuses on surface form)
6. One zero precision → zero BLEU

Alternatives:

- METEOR: Considers synonyms, stemming
- ROUGE: For summarization
- TER: Translation Edit Rate
- BERT-Score: Contextual embeddings

Question 6: Explain Ethical Issues in Machine Translation

Answer:

MT systems face significant ethical challenges, particularly regarding gender bias, cultural bias, and fairness in translation quality across languages.

1. Gender Bias in Machine Translation

Problem: MT systems often default to stereotypical gender when translating from gender-neutral languages to gendered languages.

Example: Hungarian to English

Hungarian uses gender-neutral "ő" (they/he/she). When translating:

- "ő egy pók" → **she** is a nurse
- "ő egy tudós" → **he** is a scientist
- "ő egy mérnök" → **he** is an engineer
- "ő egy tanár" → **she** is a teacher
- "ő egy vezérigazgató" → **he** is a CEO

Pattern: Male-dominated professions → "he", Female-dominated → "she"

Why This Happens:

1. Training data reflects historical stereotypes
2. Models learn correlations between professions and genders
3. Lack of context for gender inference
4. Societal biases in training data

Impact:

- Reinforces harmful stereotypes
- Limits representation and diversity
- Affects professional perceptions
- Perpetuates gender inequality

2. Non-Stereotypical Gender Roles

WinoMT Dataset tests MT on counter-stereotypical examples:

Example:

- "The doctor asked the nurse to help **her** in the operation"
- Challenge: Doctor stereotypically male, but "her" indicates female
- MT systems often mistranslate

3. Cultural and Linguistic Bias

Contextual Nuances:

- Some languages encode cultural concepts without direct equivalents
- Translation may lose cultural significance
- "Neutral" translations may favor dominant cultures

Examples:

- Japanese "おもてなし" - hospitality without English equivalent
- Arabic complex pronoun systems
- Chinese kinship terms - paternal vs. maternal

4. Low-Resource Language Inequality

Problem: Most MT focuses on high-resource languages.

Statistics:

- 7000+ languages exist
- MT available for < 200 languages
- Quality varies dramatically

Consequences:

- **Digital Divide**: Speakers of low-resource languages excluded
- **English-Centric**: Most systems use English as pivot
- **Cultural Erasure**: Minority languages underrepresented
- **Economic Impact**: Limited access to global information

English-Pivot Model:
Language A → English → Language B

Issues:

- Assumes English adequately captures meaning
- Introduces double translation errors
- Favors languages similar to English

5. High-Stakes Translation Errors

Medical Translation:

- Example: Misinterpreting "intoxicado" (poisoned) as "intoxicated" (drunk)
- Impact: Wrong diagnosis, inappropriate treatment
- Consequence: Patient harm

Legal Translation:

- Misinterpreting legal terms
- Impact: Misunderstanding of rights
- Consequence: Unjust outcomes

Humanitarian Settings:

- Mistranslating asylum testimonies
- Impact: Denied refugee status
- Consequence: Deportation to dangerous situations

6. Proposed Solutions

For Gender Bias:

1. **Balanced Training Data**
   - Include diverse gender representations
   - Counter-stereotypical examples
   - Manual curation
2. **Contextual Gender Inference**
   - Use broader context
   - Coreference resolution
   - User input for ambiguity
3. **Gender-Neutral Options**
   - Multiple translation options
   - Gender-neutral language ("they")
   - Display confidence levels
4. **Evaluation Metrics**
   - Bias-specific benchmarks
   - Counter-stereotypical performance
   - Regular fairness audits

For Low-Resource Languages:

1. **Participatory Design**
   - Engage native speakers
   - Community-driven data
   - Local experts in evaluation
2. **Technical Approaches**
   - Transfer learning
   - Multilingual models
   - Cross-lingual embeddings
3. **Infrastructure Support**
   - Research funding
   - Online communities
   - Mentoring programs

For High-Stakes Applications:

1. **Uncertainty Quantification**
   - Confidence scores
   - Highlight uncertain segments
   - Trigger human review
2. **Human-in-the-Loop**
   - Professional translators for critical domains
   - Post-editing by experts

    - Quality assurance
3. **Domain Adaptation**
    - Specialized models
    - Terminology databases
    - Context-aware translation

7. Ethical Framework

Principles:

1. **Transparency**
    - Disclose limitations
    - Explain how translations generated
    - Clear training data provenance
2. **Fairness**
    - Equitable quality across demographics
    - Address systematic biases
    - Inclusive development
3. **Accountability**
    - Clear responsibility for errors
    - Issue reporting mechanisms
    - Regular bias audits
4. **Privacy**
    - Protect sensitive information
    - User consent for data
    - Secure handling
5. **Inclusivity**
    - Support underserved communities
    - Diverse stakeholders involved
    - Global linguistic diversity

Challenges:

1. How to quantify fairness?
2. Accuracy vs. fairness trade-offs
3. Language/culture continuously evolve
4. Global coordination needed
5. Balancing commercial/social goals

Conclusion:

Ethical considerations in MT are critical for building fair systems. Addressing bias, supporting low-resource languages, and ensuring high-stakes safety requires technical innovation, community engagement, and commitment to inclusive AI development.