Comprehensive Practice Problem Set

Natural Language Processing (AIMLCZG530)

Saurabh

Exam Structure Overview

- **Coverage:** Modules 1–6 (Introduction to Neural POS Tagging)

Q1. Module 1: Introduction & Applications

Topics: Ambiguity, Preprocessing, Levels of Analysis, Evaluation.

Variation 1.1: Ambiguity Identification

Question

Consider the sentence: *"I made her duck."*

1. Identify two different types of ambiguity present in this sentence.
2. Explain the interpretations associated with each type.

Solution:

- **Syntactic/Structural Ambiguity:** The structure of the sentence allows multiple interpretations.
  - *Meaning 1:* I cooked a duck meat dish for her. (Ditransitive verb).
  - *Meaning 2:* I caused her to lower her head quickly. (Causative verb).
- **Lexical Ambiguity:** The word "duck" has multiple meanings.
  - *Meaning A:* A water bird (Noun).
  - *Meaning B:* To lower head/body (Verb).

Variation 1.2: Preprocessing Pipeline

Question

Given the raw text: *"Ph.D. students aren't going to the AI-Conference!!"* Perform the following steps sequentially and show the output after each step:

1. **Tokenization:** Split on whitespace and punctuation (keep sentence-ending punctuation separate, handle "n't").
2. **Stop Word Removal:** Remove: *to, the, is, are, n't.*
3. **Lemmatization:** Convert words to their base dictionary form.

Solution:

1. **Tokenization:** `["Ph.D.", "students", "are", "n't", "going", "to", "the", "AI-Conference", "!", "!"]`
2. **Stop Word Removal:** `["Ph.D.", "students", "are", "going", "AI-Conference", "!", "!"]` (Note: "are" is often a stop word, but if "aren't" was split, "are" might remain or be removed depending on the specific list. Assuming "are" is removed). Corrected: `["Ph.D.", "students", "going", "AI-Conference", "!", "!"]`
3. **Lemmatization:** `["Ph.D.", "student", "go", "AI-Conference", "!", "!"]`

Variation 1.3: Levels of Language Analysis

Question

Identify the specific level of linguistic analysis (Lexical, Syntactic, Semantic, or Pragmatic) where the failure occurs in the following sentences. Briefly explain why.

1. "Large have green ideas nose."
2. "Colorless green ideas sleep furiously."
3. "Can you pass the salt?" (Interpreted literally as a query about ability rather than a request).

Solution:

1. **Syntactic Level:** The sentence violates grammatical rules of word order (e.g., Adjective before Verb without Noun), even though words are lexically valid.
2. **Semantic Level:** The sentence is grammatically correct (Syntactically valid) but makes no sense in terms of meaning (ideas cannot be green or sleep).
3. **Pragmatic Level:** The literal meaning is valid, but the failure lies in understanding the speaker's intent (contextual use) as a request rather than a question of ability.

Variation 1.4: Evaluation Metrics

Question

You are evaluating a binary classification system for sentiment analysis (Positive/Negative). The system produced the following confusion matrix results:

- True Positives (TP): 80
- False Positives (FP): 20
- False Negatives (FN): 10
- True Negatives (TN): 90

Calculate the **Precision**, **Recall**, and **Accuracy** of the system.

Solution:

1. **Precision:** Ratio of correctly predicted positive observations to the total predicted positive observations.
$$\text{Precision} = \frac{TP}{TP + FP} = \frac{80}{80 + 20} = \frac{80}{100} = 0.8$$
2. **Recall:** Ratio of correctly predicted positive observations to the all observations in actual class.
$$\text{Recall} = \frac{TP}{TP + FN} = \frac{80}{80 + 10} = \frac{80}{90} \approx 0.89$$
3. **Accuracy:** Ratio of correctly predicted observations to the total observations.
$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} = \frac{80 + 90}{80 + 20 + 10 + 90} = \frac{170}{200} = 0.85$$

Q2. Module 2: Language Models

Topics: N-grams, Smoothing, Perplexity.

Variation 2.1: MLE Bigram Calculation

Question

Given a corpus of 3 sentences:

- `<s> a b a </s>`
- `<s> b a b </s>`
- `<s> a a </s>`

Calculate the Maximum Likelihood Estimation (MLE) probability for the bigram $P(b|a)$.

Solution: Formula: $P(b|a) = \frac{Count(a,b)}{Count(a)}$

- **Count(a):** occurrences of 'a' as the first word in a bigram (do not count 'a' if it is the last word before `</s>` unless considering boundary, usually we count tokens in history).
- Occurrences of 'a' in history context:
  - Sent 1: 'a' (before b), 'a' (before </s>)
  - Sent 2: 'a' (before b)
  - Sent 3: 'a' (before a), 'a' (before </s>)
- Total $Count(a) = 5$.
- **Count(ab):** 'a b' appears in Sent 1 and Sent 2. Total = 2.
- $P(b|a) = 2/5 = 0.4$.

Variation 2.2: Add-k Smoothing

Question

Using the same corpus as above, calculate the probability $P(b|a)$ using **Add-1 Smoothing**. Assume vocabulary size $V = 4$ (tokens: a, b, <s>, </s>).

Solution: Formula: $P^*(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)+1}{C(w_{i-1})+V}$

- $C(ab) = 2$
- $C(a) = 5$
- $V = 4$
- $P^*(b|a) = \frac{2+1}{5+4} = \frac{3}{9} \approx 0.33$.

Variation 2.3: Perplexity Calculation

Question

A language model assigns the following probabilities to the words in a test sentence $W = w_1 w_2 w_3$:

- $P(w_1) = 0.5$
- $P(w_2|w_1) = 0.2$
- $P(w_3|w_1, w_2) = 0.1$

Calculate the Perplexity (PP) of this sentence.

Solution:

- Probability of sentence $P(S) = 0.5 \times 0.2 \times 0.1 = 0.01$.
- $N = 3$.
- $PP(S) = P(S)^{-1/N} = (0.01)^{-1/3} = (10^{-2})^{-1/3} = 10^{2/3} \approx 4.64$.

Variation 2.4: Linear Interpolation

Question

Calculate the interpolated trigram probability $\hat{P}(w_3|w_1 w_2)$ using $\lambda_1 = 0.5, \lambda_2 = 0.3, \lambda_3 = 0.2$. Given:

- Unigram $P(w_3) = 0.01$
- Bigram $P(w_3|w_2) = 0.05$
- Trigram $P(w_3|w_1 w_2) = 0.2$

Solution:

$$\hat{P} = \lambda_1 P_{tri} + \lambda_2 P_{bi} + \lambda_3 P_{uni}$$

$$\hat{P} = (0.5 \times 0.2) + (0.3 \times 0.05) + (0.2 \times 0.01)$$

$$\hat{P} = 0.1 + 0.015 + 0.002 = 0.117$$

Q3. Module 3: Neural LMs & Intro to LLM

Topics: NN Architecture, Activation, Softmax, Loss, Parameters, LLM Basics.

Variation 3.1: Architecture Diagram Interpretation

Question

In a Feedforward Neural Language Model (Bengio et al.):

1. If the vocabulary size is $|V| = 5000$ and the projection (embedding) dimension is $d = 100$, what is the size of the embedding matrix C?
2. If the context window considers 3 previous words, what is the dimension of the input vector x fed into the hidden layer (after concatenation of embeddings)?

Solution:

1. Matrix Size: $V \times d = 5000 \times 100$.
2. Input Vector Size: $3 \times d = 3 \times 100 = 300$.

Variation 3.2: Activation Function Calculation

Question

A neuron receives an input $z = 0.0$. Calculate the output activation for:

1. Sigmoid function $\sigma(z)$.
2. Tanh function $\tanh(z)$.

Solution:

1. Sigmoid: $\sigma(0) = \frac{1}{1+e^{-0}} = \frac{1}{1+1} = 0.5$.
2. Tanh: $\tanh(0) = \frac{e^0 - e^{-0}}{e^0 + e^{-0}} = \frac{1-1}{1+1} = 0$.

Variation 3.3: Softmax Output Calculation

Question

The output layer of a Neural Language Model produces the raw logits vector $z = [2.0, 1.0, 0.1]$ for a vocabulary of 3 words (Word A, Word B, Word C). Calculate the Softmax probability for **Word A**. Use $e \approx 2.7$.

Solution: Formula: $\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$

- $e^{2.0} \approx 7.29$
- $e^{1.0} \approx 2.7$
- $e^{0.1} \approx 1.1$
- Sum $= 7.29 + 2.7 + 1.1 = 11.09$
- $P(\text{Word A}) = \frac{7.29}{11.09} \approx 0.657$

Variation 3.4: Parameter Counting

Question

Calculate the total number of trainable parameters (weights) in the following Feedforward Neural LM:

- Vocabulary size (V): 100
- Embedding dimension (d): 10
- Context window size (N): 2 words
- Hidden layer neurons (H): 50
- Bias used in hidden layer and output layer.

Hint: Include Embedding Matrix ($V \times d$), Hidden Weights ($(N \times d) \times H + H$), and Output Weights ($H \times V + V$).

Solution:

1. **Embedding Matrix:** $100 \times 10 = 1000$.
2. **Hidden Layer Weights:** Input size is 2 words $\times$ 10 dim $= 20$. Weights: $20 \times 50 = 1000$. Biases: 50. Total: 1050.
3. **Output Layer Weights:** Connects hidden (50) to vocab (100). Weights: $50 \times 100 = 5000$. Biases: 100. Total: 5100.
4. **Total:** $1000 + 1050 + 5100 = 7150$ parameters.

Variation 3.5: Cross-Entropy Loss

Question

A Neural Language Model predicts the following probabilities for the next word in a sequence:

- $P(\text{apple}) = 0.1$
- $P(\text{banana}) = 0.6$
- $P(\text{cherry}) = 0.3$

If the actual next word in the training data is **"apple"**, calculate the Cross-Entropy Loss. *Formula:* $L = -\sum y_i \log(\hat{y}_i)$. (Use $\log_e$ or $\log_2$ as per standard convention, typically natural log).

Solution:

- The true distribution y is one-hot for "apple": $[1, 0, 0]$.
- The predicted distribution $\hat{y}$ is: $[0.1, 0.6, 0.3]$.
- $L = -[1 \times \log(0.1) + 0 \times \log(0.6) + 0 \times \log(0.3)]$
- $L = -\log(0.1)$
- If using base e: $-\ln(0.1) \approx 2.30$. (If base 2: ≈ 3.32).

Variation 3.6: XOR Problem & Non-linearity

Question

Why is a single-layer Perceptron (without a hidden layer) unable to solve the XOR problem? How does adding a hidden layer with non-linear activation functions solve this?

Solution:

- **Reason:** The XOR function is not **linearly separable**. A single perceptron can only define a linear decision boundary (a straight line) which cannot separate the XOR outputs (0,0 $\rightarrow$ 0; 1,1 $\rightarrow$ 0 vs 0,1 $\rightarrow$ 1; 1,0 $\rightarrow$ 1).
- **Solution:** Adding a hidden layer allows the network to learn intermediate representations (features) that essentially transform the input space, making the classes linearly separable in the higher-dimensional hidden space. Non-linear activations are crucial because multiple linear layers collapse into a single linear layer.

Variation 3.7: Prompt Engineering Concepts

Question

1. Distinguish between **Pre-training** and **Fine-tuning** in the context of Large Language Models (LLMs). 2. In **Prompt Engineering**, what is the difference between **Zero-shot** and **Few-shot** inference? Provide a brief example of Few-shot.

Solution: 1. **Pre-training:** Training the model on a massive dataset to learn general language patterns (next word prediction). **Fine-tuning:** Training the pre-trained model on a smaller, specific dataset to specialize it for a task (e.g., medical diagnosis). 2. **Difference:** Zero-shot gives the model a task with no examples. Few-shot provides a few examples of input-output pairs before asking the question. * *Example Few-shot:* * "Translate English to French." * "Hello -> Bonjour" * "Dog -> Chien" * "Cat -> " (Model predicts: Chat)

Q4. Module 4: Vector Semantics

Topics: TF-IDF, Cosine Similarity, Co-occurrence.

Variation 4.1: TF-IDF Calculation

Question

Consider a corpus of $N = 100$ documents. The word "neural" appears in 10 documents. In Document D_1, "neural" appears 5 times. Calculate the TF-IDF weight for "neural" in D_1. Use log-normalization for TF $(1 + \log_{10}(f))$ and standard IDF $(\log_{10}(N/df))$.

Solution:

- **TF:** $1 + \log_{10}(5) \approx 1 + 0.7 = 1.7$.
- **IDF:** $\log_{10}(100/10) = \log_{10}(10) = 1$.
- **TF-IDF:** $1.7 \times 1 = 1.7$.

Variation 4.2: Cosine Similarity

Question

Given two word vectors $\vec{A} = [1, 2, 0]$ and $\vec{B} = [0, 3, 4]$. Calculate the Cosine Similarity between them.

Solution: Formula: $\frac{\vec{A} \cdot \vec{B}}{||\vec{A}|| \times ||\vec{B}||}$

- Dot Product: $(1)(0) + (2)(3) + (0)(4) = 6$.
- $||\vec{A}|| = \sqrt{1^2 + 2^2 + 0} = \sqrt{5}$.
- $||\vec{B}|| = \sqrt{0 + 3^2 + 4^2} = \sqrt{9 + 16} = \sqrt{25} = 5$.
- Cosine Sim $= \frac{6}{5\sqrt{5}} \approx \frac{6}{11.18} \approx 0.536$.

Variation 4.3: Document Vector (Centroid)

Question

A document consists of two words: "Apple" and "Red". Embeddings are:

- Apple: $[0.5, 0.5]$
- Red: $[0.1, 0.9]$

Calculate the document embedding vector using the **Centroid Method** (Average).

Solution:

$$\vec{D} = \frac{\vec{v}_{apple} + \vec{v}_{red}}{2} = \frac{[0.5 + 0.1, 0.5 + 0.9]}{2} = \frac{[0.6, 1.4]}{2} = [0.3, 0.7]$$

Variation 4.4: Euclidean Distance

Question

Calculate the Euclidean distance between the vectors $\vec{A} = [1, 5]$ and $\vec{B} = [4, 1]$.

Solution:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$d = \sqrt{(4-1)^2 + (1-5)^2} = \sqrt{3^2 + (-4)^2} = \sqrt{9 + 16} = \sqrt{25} = 5$$

Q5. Module 4: Word Embedding

Topics: Word2Vec (Skip-gram, CBOW), Updates, Analogies.

Variation 5.1: Skip-gram Architecture

Question

You are training a Skip-gram model with a window size of 2 ($C = 2$). Given the sentence: *"The quick brown fox jumps"*. If the target word is **"brown"**, identify the input word and the list of target output (context) words for training pairs.

Solution:

- **Input Word:** "brown"
- **Window:** ± 2 words.
- **Context Words:** "The", "quick", "fox", "jumps".
- **Training Pairs:** (brown, The), (brown, quick), (brown, fox), (brown, jumps).

Variation 5.2: Word2Vec Backward Propagation

Question

Assume a Skip-gram model with Negative Sampling.

- Target: "cat", Context: "meow" (Positive pair, $y = 1$).
- Input Vector $\vec{v}_{cat} = [0.5, 0.5]$.
- Output Vector $\vec{u}_{meow} = [1.0, 0.0]$.
- The model predicts probability $P = \sigma(\vec{v} \cdot \vec{u}) = 0.62$.
- Learning Rate $\eta = 0.1$.

Calculate the updated input vector $\vec{v}_{cat}^{new}$.

Solution:

- Error term $EI = P - y = 0.62 - 1 = -0.38$.
- Gradient w.r.t input: $EI \times \vec{u}_{meow} = -0.38 \times [1.0, 0.0] = [-0.38, 0.0]$.
- Update Rule: $\vec{v}^{new} = \vec{v}^{old} - \eta \times \text{Gradient}$.
- $\vec{v}^{new} = [0.5, 0.5] - 0.1 \times [-0.38, 0.0]$.
- $\vec{v}^{new} = [0.5, 0.5] - [-0.038, 0.0] = [0.538, 0.5]$.

Variation 5.3: Word Analogy

Question

Using the "parallelogram" model for analogies, how would you calculate the vector for "King" given vectors for "Queen", "Man", and "Woman"? (Analogy: Woman is to Queen as Man is to King).

Solution:

$$\vec{v}_{King} \approx \vec{v}_{Queen} - \vec{v}_{Woman} + \vec{v}_{Man}$$

Variation 5.4: Count vs Prediction Based

Question

Compare **LSA (Latent Semantic Analysis)** and **Word2Vec** on two parameters:

1. Sparsity of the resulting vectors.
2. Interpretability of dimensions.

Solution:

1. **Sparsity:** LSA (SVD on count matrix) produces dense vectors. Word2Vec also produces dense vectors. (Note: Raw TF-IDF is sparse, LSA is dense).
2. **Interpretability:** LSA dimensions correspond to orthogonal axes of variance (latent topics). Word2Vec dimensions are generally not interpretable individually; meaning is distributed.

Q6. Module 5: POS Tagging - HMM

Topics: Transition/Emission Probs, Disambiguation.

Variation 6.1: Transition Probability

Question

In a tagged corpus, the tag **DT** (Determiner) appears 100 times. It is followed by **NN** (Noun) 60 times, **JJ** (Adjective) 30 times, and **VB** (Verb) 10 times. Calculate the transition probabilities $P(NN|DT)$ and $P(JJ|DT)$.

Solution:

- $P(NN|DT) = \frac{Count(DT,NN)}{Count(DT)} = \frac{60}{100} = 0.6$.
- $P(JJ|DT) = \frac{Count(DT,JJ)}{Count(DT)} = \frac{30}{100} = 0.3$.

Variation 6.2: HMM Disambiguation

Question

Disambiguate the word **"book"** in the sequence *"read book"*.

- Previous tag for "read": **VB**.
- Possible tags for "book": **NN**, **VB**.
- Transitions: $P(NN|VB) = 0.4$, $P(VB|VB) = 0.1$.
- Emissions: $P(\text{"}book\text{"}|NN) = 0.05$, $P(\text{"}book\text{"}|VB) = 0.01$.

Which tag is chosen?

Solution: Calculate Score = Transition $\times$ Emission.

- **Case NN:** $0.4 \times 0.05 = 0.020$.
- **Case VB:** $0.1 \times 0.01 = 0.001$.

- **Decision:** $0.020 > 0.001$, so "book" is tagged as **NN**.

Variation 6.3: Hidden vs Observed

Question

In the context of HMM POS Tagging:

1. What constitutes the **Hidden States**?
2. What constitutes the **Observations**?

Solution:

1. **Hidden States:** The Parts of Speech tags (Noun, Verb, etc.).
2. **Observations:** The actual words in the sentence.

Variation 6.4: Decoding Problem

Question

Which algorithm is used to find the most probable sequence of hidden states (tags) given a sequence of observations (words) in an HMM? What is its time complexity regarding the number of states N?

Solution:

- **Algorithm:** Viterbi Algorithm.
- **Complexity:** $O(N^2 \cdot T)$, where N is states and T is sequence length.

Q7. Module 6: Statistical, ML & Neural POS Models

Topics: Viterbi Trellis, MEMM, Neural Tagging.

Variation 7.1: Viterbi Trellis Calculation

Question

Calculate the Viterbi path probability for the first word in the sentence *"Time flies"*.

- States: **N**, **V**.
- Start Probs: $\pi(N) = 0.8$, $\pi(V) = 0.2$.
- Emission Probs for "Time": $P(\text{"}Time\text{"}|N) = 0.5$, $P(\text{"}Time\text{"}|V) = 0.1$.

Find $V_1(N)$ and $V_1(V)$.

Solution:

- $V_1(N) = \pi(N) \times P(\text{"}Time\text{"}|N) = 0.8 \times 0.5 = 0.4$.
- $V_1(V) = \pi(V) \times P(\text{"}Time\text{"}|V) = 0.2 \times 0.1 = 0.02$.
- Best current tag: **N**.

Variation 7.2: Viterbi Backtrace

Question

You are at the end of a sentence of length $T = 3$. The Viterbi values at $T = 3$ are:

- State N: Value=0.005, Backpointer=State V.
- State V: Value=0.001, Backpointer=State N.

Backpointers at $T = 2$:

- If current is N, prev was D.
- If current is V, prev was N.

Determine the best tag sequence.

Solution:

1. **Last step ($T = 3$):** Max value is 0.005 (State N). Tag 3 = N.
2. **Step $T = 2$:** Backpointer of Tag 3 (N) points to V. Tag 2 = V.
3. **Step $T = 1$:** Backpointer of Tag 2 (V) points to N. Tag 1 = N.
4. **Sequence:** N $\rightarrow$ V $\rightarrow$ N.

Variation 7.3: MEMM vs HMM

Question

Why is the Maximum Entropy Markov Model (MEMM) generally considered more flexible than a standard HMM for POS tagging? Provide one mathematical or architectural reason.

Solution: MEMM is a **discriminative** model ($P(Tag|Word)$) while HMM is **generative** ($P(Word, Tag)$). This allows MEMM to utilize arbitrary, overlapping features (like capitalization, suffixes, previous words, next words) without violating independence assumptions that limit HMMs.

Variation 7.4: Neural POS Tagging (RNN)

Question

In a simple RNN-based POS tagger, the input is a sequence of word embeddings. What is the dimension of the output layer at each time step t, and which activation function is applied to it to get tag probabilities?

Solution:

- **Dimension:** The size of the Tagset (number of unique POS tags).
- **Activation:** Softmax (to produce a probability distribution over the tags).