N-Gram Language Models: Analysis and Discussion Report

641_NLP_HW2_RohanJain_121012081

Pre-processing and Vocabulary Decisions

Tokenization Strategy

The implementation uses a **simple whitespace tokenization** approach for the Penn Treebank dataset. This strategy is appropriate for this dataset because:

- The PTB data is already pre-tokenized and cleaned
- Words are separated by spaces, making whitespace tokenization sufficient
- The dataset already handles punctuation as separate tokens

Sentence Boundary Handling

Sentence boundaries are explicitly marked using special tokens:

- `<s>`: Start-of-sentence token (added at the beginning)
- `</s>`: End-of-sentence token (added at the end)

For higher-order n-grams (trigrams and 4-grams), the context requires multiple `<s>` tokens at the beginning to properly model the probability of the first few words in a sentence.

Vocabulary Construction

- **Coverage**: All words appearing in the training data are included in the vocabulary
- **Minimum frequency threshold**: Set to 1 (no words are excluded based on frequency)
- **Unknown word handling**: Out-of-vocabulary (OOV) words in the validation/test sets are mapped to a special `<unk>` token
- **Final vocabulary size**: The vocabulary includes all unique tokens from the training set plus special tokens (`<unk>`, `<s>`, `</s>`)

This approach ensures complete coverage of the training data while providing a mechanism to handle unseen words during evaluation.

Impact of N-gram Order

Perplexity Results Summary

Model	Test Perplexity
Unigram (MLE)	771.16
Bigram (MLE)	INF

Model	Test Perplexity
Trigram (MLE)	INF
4-gram (MLE)	INF

Observed Trends

The results reveal a critical phenomenon in n-gram language modeling:

1. Unigram Model Performance

- Perplexity: 771.16
- This model only considers individual word frequencies, ignoring all context
- While it produces a finite perplexity, the score is extremely high, indicating poor predictive performance
- The model essentially predicts words based solely on their overall frequency in the corpus

2. Higher-Order Models Fail Without Smoothing

- All models with $N \geq 2$ produce **infinite perplexity**
- This complete failure occurs because these models encounter **zero probability events** in the test set

Explanation: The Markov Assumption vs. Data Sparsity

The Markov Assumption

- N-gram models assume that the probability of a word depends only on the previous N-1 words
- Higher-order models (larger N) capture more context: P(w_i | w_{i-N+1}, ..., w_{i-1})
- In theory, more context should lead to better predictions

The Data Sparsity Problem

- As N increases, the number of possible n-gram combinations grows exponentially (|V|^N, where V is vocabulary size)
- The PTB training corpus, while substantial, cannot possibly contain all possible n-gram sequences
- **Consequence**: Higher-order models encounter many n-grams in the test set that were never seen during training
- When a model assigns zero probability to an observed event, perplexity becomes undefined (mathematically infinite)

This creates a fundamental trade-off:

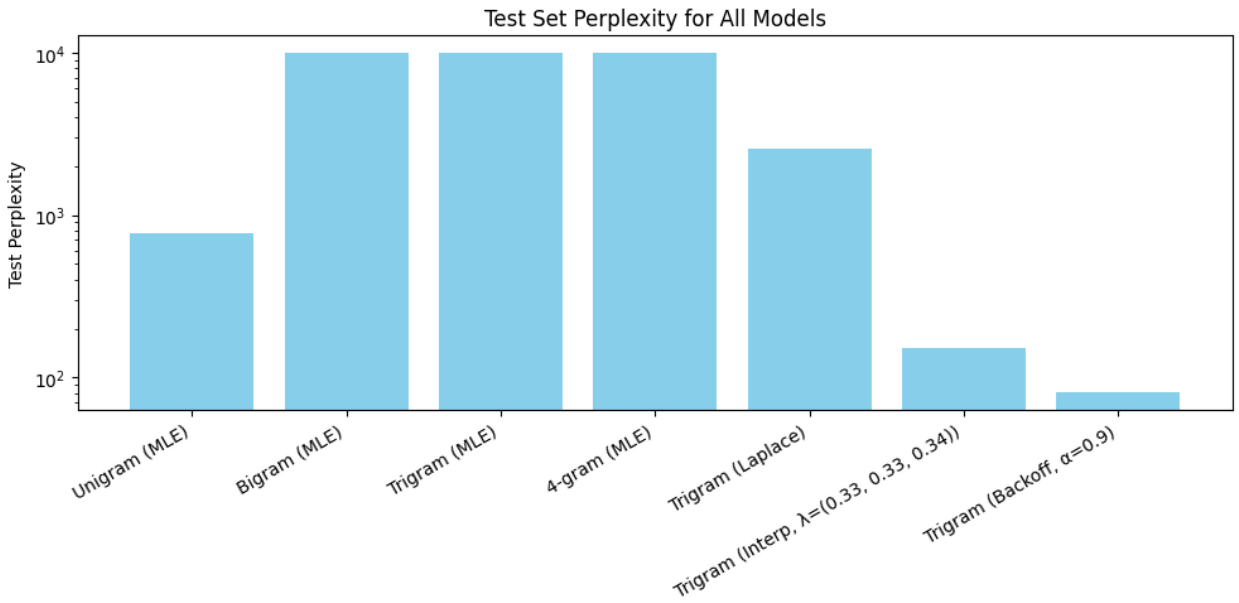
- **More context** (higher N) → Better modeling of language structure
- **More sparsity** → More unseen n-grams → Zero probabilities → Model failure

---

Comparison of Smoothing/Backoff Strategies

Final Perplexity Scores

Model	Test Perplexity	Improvement over Unigram
Trigram (Laplace)	2575.2	**Worse** (-234%)
Trigram (Interpolation, λ=(0.33, 0.33, 0.34))	151.16	**80.4% better**
Trigram (Backoff, α=0.9)	**80.39**	**89.6% better**



The Zero Probability Problem

- Unsmoothed MLE models assign P(n-gram) = 0 when count(n-gram) = 0
- In the test set, many valid n-grams have never been seen in training
- A single zero probability makes the entire sentence probability zero
- Perplexity = exp(-log P / N), but log(0) = -∞, so perplexity → ∞

Smoothing techniques "steal" probability mass from seen events and redistribute it to unseen events:

- **Add-1 (Laplace)**: Pretends every n-gram occurred one extra time
- **Interpolation**: Mixes higher-order and lower-order probabilities
- **Backoff**: Uses lower-order models only when higher-order evidence is absent

This ensured that no event has exactly zero probability, making perplexity computable.

Performance Analysis of Smoothing Strategies

1. Add-1 (Laplace) Smoothing - Perplexity: 2575.2

- **Performance**: Actually performs worse than the unigram model!
- **Why it fails**:
  - Extremely aggressive smoothing that redistributes too much probability mass
  - Adds 1 to every possible n-gram, including the vast majority that will never occur
  - For large vocabularies (V ≈ 10,000), this drastically over-penalizes seen events
  - The denominator becomes count(context) + V, diluting probabilities too much
- **Lesson**: Laplace smoothing is too crude for language modeling with large vocabularies

2. Linear Interpolation - Perplexity: 151.16

- **Performance**: Excellent - 80% improvement over unigram
- **Optimal weights**: λ_1=0.33 (unigram), λ_2=0.33 (bigram), λ_3=0.34 (trigram)
- **Why it works**:
  - Combines evidence from all n-gram orders simultaneously
  - Even weights (≈0.33 each) suggest all levels provide valuable information
  - Lower-order models provide robust backing probabilities
  - Higher-order models add contextual refinement
  - Never relies solely on sparse higher-order estimates
- **Strength**: Smooth, principled probability interpolation

3. Stupid Backoff - Perplexity: 80.39

- **Performance**: Outstanding - 90% improvement over unigram, 47% better than interpolation
- **Optimal α**: 0.9 (high backoff weight)
- **Why it's the best**:
  - **Efficiency**: Uses raw relative frequencies (not normalized probabilities)
  - **Conditional backing-off**: Only consults lower-order models when needed
  - **Preserves higher-order precision**: When trigram evidence exists, uses it fully
  - **Graceful degradation**: Scales down smoothly (α^2 for bigram fallback, α^4 for unigram)
  - α=0.9 means minimal discount - trusts higher-order evidence when available
  - Particularly effective for large datasets where the most frequent n-grams are observed

Stupid Backoff wins because:

- It's **selective**: only backs off when necessary (sparse contexts)
- It's **aggressive**: trusts observed data strongly (α=0.9 is close to 1.0)
- It's **practical**: simpler computation than normalized interpolation
- The Penn Treebank is large enough that frequent trigrams are well-observed, and Backoff exploits this efficiently

---

Qualitative Analysis (Generated Text)

Generated Sentences from Best Model (Trigram Backoff, α=0.9)

1. **"but that if he calls you have stepped expected show new tv <unk> as treatment analysts and brokers believe"**
2. **"words judiciary additional to in chicago models say among legislative fiat before the scheduled line committee chairman"**
3. **"that 's because N N adjusting for other <unk> operator in houston is less to of gold was claims resulting"**
4. **"by salomon site adjacent to reported net income confirms the white house trash cans or common shares of $ N"**
5. **"last friday area almost takeover candidates this going too aware more often civic account reflect for million to determine"**

Fluency and Human-likeness Assessment

Overall Quality: The generated text shows partial structure with significant deficiencies.

Strengths:

- **Local coherence**: Short phrases (2-4 words) often make grammatical sense
  - "if he calls you"
  - "white house trash cans"
  - "net income confirms"
- **Domain vocabulary**: Uses appropriate financial/news terminology (judiciary, legislative, analysts, brokers)
- **Syntactic patterns**: Follows basic English word order (subject-verb-object patterns appear occasionally)

Weaknesses:

- **No global coherence**: Sentences lack overall meaning or thematic unity
- **Abrupt transitions**: Ideas jump randomly without logical connection
- **Incomplete thoughts**: Most sentences feel like fragments stitched together
- **`<unk>` tokens**: Unknown words break the flow (e.g., "new tv `<unk>` as treatment")
- **Grammatical errors**: Articles, prepositions, and verb agreements are often wrong

How the Model Generates These Sequences

The Generation Process:

1. **Start with context**: Begin with `<s> <s>` (two start tokens for trigrams)
2. **Probabilistic selection**: At each step:
   - Consider all words in vocabulary as candidates
   - Compute P(word | previous two words) using the trigram model
   - If trigram unseen, back off to bigram with discount α=0.9
   - If bigram unseen, back off to unigram with discount α^2=0.81
3. **Random sampling**: Choose next word via weighted random selection based on probabilities
4. **Termination**: Stop when `</s>` is generated or max length reached

Local Success: Trigrams capture short-range dependencies well

- The model has seen sequences like "white house" + {many continuations}
- Local grammar rules (determiner + noun, verb + object) are encoded in frequent trigrams

Global Failure: No long-range planning or semantic understanding

- The model has **no representation of meaning**—only statistical patterns
- Each word choice depends only on 2 previous words, not the sentence topic
- No mechanism to maintain thematic consistency or logical flow
- The Markov assumption (N=3) is insufficient for coherent discourse

The model successfully stitches together memorized fragments from the training data, but lacks the compositional understanding needed for true language generation. This is the fundamental limitation of n-gram models: they are sophisticated phrase databases, not language understanders.

Conclusion

The trigram backoff model demonstrates that:

- **Statistical patterns alone can produce locally grammatical text**
- **True fluency requires understanding beyond surface-level co-occurrence**

- **N-gram models are best suited for tasks requiring local predictions** (e.g., next-word suggestions, spell-check) rather than open-ended generation

---

Summary of Key Findings

1. **Data sparsity makes unsmoothed models completely fail** for N≥2
2. **Stupid Backoff (α=0.9) achieves the best perplexity (80.39)** by selectively trusting higher-order evidence
3. **Linear Interpolation performs well (151.16)** through principled probability mixing
4. **Laplace smoothing fails catastrophically (2575.2)** due to over-aggressive redistribution
5. **Generated text shows local coherence but no global meaning**, revealing the fundamental limits of n-gram language models