Language Modeling for Roman Urdu Diaries

June 22, 2025

1 Implementation Process

I implemented unigram, bigram, and trigram models with different strategies. These included random selection, max probability selection, backoff, and interpolation. Additionally, I explored advanced models like backward bigram and bidirectional bigram for better text generation.

Backoff helped when a higher-order n-gram was missing, using lower-order n-grams instead. Fixed lambda interpolation blended different n-grams with a weighted sum.

To ensure smooth transitions between sentences, if not the first sentence, the last word or next predicted of last word (or bigram) of the previous sentence was used as the starting point for the next one. This prevented abrupt changes and made the text flow more naturally.

2 Comparison of Models

2.1 Unigram Models

Unigram (Random Selection) → Picks words based on unigram probabilities. No sentence structure, leading to randomness.
Unigram (Max Selection) → Always picks the most frequent word, leading to extreme repetition (e.g., *aur aur aur*).

2.2 Bigram Models

Bigram (Random Selection) → Generates sentences using bigrams but can still be unpredictable.
Bigram (Max Selection) → Uses the most probable next word. However, it often repeats the same phrase (e.g., *aur hamne khana khaya* in almost every sentence).
Bigram (Random + Backoff) → Uses bigram probabilities but backs off to unigram if no valid bigram exists. This prevents stopping but introduces randomness.
Bigram (Max + Backoff) → Uses the most probable bigram but falls back to unigram if needed. Reduces phrase repetition but still struggles with generic outputs.
Bigram (Random + Backoff + Fixed Lambda Interpolation) → Blends bigram and unigram probabilities using a fixed weight. Produces more varied text.
Bigram (Max + Backoff + Fixed Lambda Interpolation) → Uses max probability bigram + unigram interpolation. More structured than pure bigram but still limited by frequent words.

2.3 Trigram Models

Trigram (Random Selection) → Picks the next word randomly from the trigram distribution. More structure than bigrams but still unpredictable.
Trigram (Max Selection) → Picks the most probable trigram, reducing randomness but sometimes causing phrase repetition.
Trigram (Random + Backoff) → Falls back to bigram or unigram when needed. Helps prevent sentence stopping.
Trigram (Max + Backoff) → Uses max probability trigram but backs off to lower n-grams when needed. More structured than pure trigram.
Trigram (Random + Backoff + Fixed Lambda Interpolation) → Uses a weighted combination of trigram, bigram, and unigram probabilities, balancing fluency and diversity.
Trigram (Max + Backoff + Fixed Lambda Interpolation) → Uses max probability trigram + bigram + unigram interpolation. Produces the best structure with minimal repetition.

2.4 Advanced Models

Backward Bigram Model → Generates text right to left instead of left to right. Changes sentence flow but harder to evaluate.
Bidirectional Bigram Model (Random Selection) → Combines forward and backward bigram contexts. More balanced but can be inconsistent.
Bidirectional Bigram Model (Max Selection) → Uses max probability bigram in both directions. Produces the most coherent sentences.

3 Challenges Faced

1. Repetition in Max Models: Max selection led to loops, repeating words or phrases (e.g., *aur aur aur* in unigram, or *aur hamne khana khaya* in bigram).
2. Sentence Stopping: If a higher n-gram was missing, the sentence often stopped early. Backoff helped, but some outputs still felt incomplete.
3. Randomness in Sampling: Random models generated varied text, but often lacked meaningful structure.
4. Data Sparsity in Trigrams: Many trigrams did not exist in the corpus, requiring frequent backoff to bigrams or unigrams.
5. Merging in Bidirectional Models: Combining left-to-right and right-to-left predictions was tricky, sometimes produced conflicting outputs.

4 Sample Outputs

Unigram (Max Selection): *Aur aur aur aur aur.* (Extreme repetition)
Bigram (Max Selection): *Aur hamne khana khaya. Aur hamne khana khaya.* (Phrase repetition)
Bigram (Random + Backoff): *Hamne aj uni jana.* (More variation, but sometimes incomplete)
Trigram (Max + Backoff + Interpolation): *Hamne uni jana tha magar mausam kharab tha.* (Best fluency)
Bidirectional Bigram (Max Selection): *Hamne aj bazar se cheezen khareedi aur ghar wapas aye.* (Most natural)

5 Perplexity Evaluation

To evaluate model effectiveness, we calculated **perplexity** on a test set using unigram, bigram, and trigram probabilities.

5.1 Formula

Perplexity is calculated as:

$$PP(W) = e^{-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i|\text{context})} \tag{1}$$

where $P(w_i|\text{context})$ is derived from unigram, bigram, or trigram models.

5.2 Implementation

We computed log probabilities for the test data:

- **Unigram Perplexity:** Uses $P(w_i)$ directly.
- **Bigram Perplexity:** Uses $P(w_i|w_{i-1})$, with backoff to unigram.
- **Trigram Perplexity:** Uses $P(w_i|w_{i-2}, w_{i-1})$, with backoff to bigram/unigram.

5.3 Results

- **Unigram Perplexity: High** (worst performance, random word selection).
- **Bigram Perplexity: Lower than unigram** (improved fluency).
- **Trigram Perplexity: Lowest** (best model, but data sparsity remains a challenge).

6 Conclusion

Higher-order n-grams improved fluency, but data sparsity caused challenges. Backoff and interpolation helped make models more robust. Trigram (Max + Backoff + Interpolation) produced the best balance of fluency and structure. The bidirectional model showed promise for more natural text generation.