

# CS6320 Assignment 1

[https://github.com/gadarsh043/NLP\\_N-gramsmodel.git](https://github.com/gadarsh043/NLP_N-gramsmodel.git)

## Group 6

### 1. TEAM MEMBERS

- Adarsh Gella: AXG240019
- Akhila Susarla: AXS240035
- Vijaya Sai Latha Pulipati: VXP230093
- Abhiram Reddy Madapa: AXM240036

### OBJECTIVE:

In this assignment, we focused on developing an n-gram-based language model using hotel reviews. We created both unigram and bigram models, experimenting with different smoothing techniques and methods for handling unknown words. Our approach involved exploring various design strategies to achieve the desired outcomes. We followed the standard procedures for model creation and testing that are typical in Natural Language Processing.

### 2. Dataset

For this assignment, we utilized a corpus of hotel reviews from Chicago establishments. The dataset consists of hotel reviews where each line represents a single review. The reviews are already tokenized and separated by spaces, making it straightforward to process. We used train.txt for training our models and validation.txt for testing and evaluation purposes.

### 3 Implementation Details

#### 3.1 Preprocessing:

Before training the unigram and bigram models, we preprocess the data by

converting all words to lowercase, removing punctuation using regular expressions, and applying lemmatization to convert words to their root forms using the NLTK WordNetLemmatizer. The tokenization was already done as the data was space-separated.

### 3.2 Unigram and Bigram Models:

We implemented both unigram and bigram models. For the unigram model, the probability of each word is independent of the previous word, whereas the bigram model calculates the probability of a word given the preceding word.

Unigram probability:  $P(\text{word}) = \text{ntrain}(\text{word}) / \text{Ntrain}$

Bigram probability:  $P(\text{word} \mid \text{previous word}) = \text{ntrain}(\text{previous word}, \text{word}) / \text{ntrain}(\text{previous word})$

### 3.3 Smoothing

We implemented the Add-k smoothing technique. Smoothing prevents zero probabilities for unseen words by modifying word counts during probability computation.

Add-k Smoothing:  $P_{\text{Add-k}}(\text{word} \mid \text{previous word}) = \frac{C(\text{previous word}, \text{word}) + k}{C(\text{previous word}) + kV}$

### 3.4 Unknown Word Handling

Unknown words are handled by replacing words that appear less than a specified frequency threshold with a special token, <UNK>. We experimented with different thresholds (1, 2, 3) where words appearing less than the threshold number of times in the training corpus are replaced with <UNK>. This helps improve model generalization on unseen data and prevents the model from being confused by rare words.

### 3.5 Perplexity

Perplexity is used to evaluate the model's performance, quantifying the uncertainty in predictions. The formula for perplexity is:

$$PP = \exp(-1/N * \sum \log P(w_i | w_{i-1}, \dots, w_{i-n+1}))$$

## 4 Evaluation, Analysis, and Findings

In this section, we present the evaluation of the model using N-gram-based methods, showing the results for different unknown word thresholds and smoothing values on both Validation and Test sets.

Unknown Word Threshold:

1 (No unknown words) Validation Set - Unigram: 58.13, Bigram: 1.77 Test Set - Unigram: 32417.82, Bigram: 259289385.13

Unknown Word Threshold:

2 Validation Set - Unigram: 1.93, Bigram: 2.55 Test Set - Unigram: 3.03, Bigram: 2.73

Unknown Word Threshold:

3 Validation Set - Unigram: 1.50, Bigram: 1.89 Test Set - Unigram: 2.24, Bigram: 2.13

Table 1:

Evaluation results showing perplexity scores for different unknown word thresholds and smoothing values.

Our results show significant improvements when implementing unknown word handling. Without unknown word handling (threshold 1), the perplexity scores are extremely high on the test set, especially for bigram models, because the models encounter many unseen words and bigrams. When we implemented unknown word handling with thresholds 2 and 3, we observed dramatic improvements in perplexity scores. The best performance was achieved with unknown word threshold 3 and no smoothing ( $k=0$ ), achieving perplexity scores of 1.50 and 2.24 for unigram and bigram models respectively on the test set. This demonstrates that proper unknown word handling is crucial for model generalization and that bigram models generally perform better than unigram models when properly implemented.

## 5 Libraries Used

- from nltk.stem import WordNetLemmatizer - to
- from sklearn.model\_selection import train\_test\_split - to split the training data into training and validation sets.
- import re - to remove punctuations from the text.
- import math - to use logarithm and power functions.
- import itertools - to flatten the tokens generated during processing.
- from collections import defaultdict, Counter - for efficient counting and data structures.

## 6 Individual Contributions

- Adarsh Gella: Code implementation, unknown word handling, and model evaluation.
- Akhila Susarla: Preprocessing, smoothing techniques, and report writing.
- Vijaya Sai Latha Pulipati: Bigram model implementation.
- Abhiram Reddy Madapa: Perplexity Calculation, Error analysis, testing, and report writing.

## 7. FEEDBACK

We found the project to be medium level. Initially we found it easy to implement but we have seen the perplexity scores were too high without proper unknown word handling. Later we invested significant time to enhance the model by concentrating more on preprocessing steps and unknown word handling techniques. Though the smoothing techniques are straight from the textbook we took our time to learn the math behind it and implement. We found the project to be interesting because it challenged our understanding of language modeling and the importance of proper data preprocessing.

## 8. References

1. <https://web.cs.hacettepe.edu.tr/~ilyas/Courses/CMP711/lec03-Langu>

[ageModels.pdf](#)

2. [https://jofrhwld.github.io/teaching/courses/2022\\_lin517/lectures/ngram/02\\_smoothing.html](https://jofrhwld.github.io/teaching/courses/2022_lin517/lectures/ngram/02_smoothing.html)
3. NLTK Documentation <https://www.nltk.org>
4. Jurafsky & Martin - Speech and Language Processing,  
Chapter 3: N-gram Language Models