N - gram based word predictor

Disha Suhas Kadam - 612303076.
Hakimuddin Slatewala - 612303065.
Jyotika Sharan - 612303075.

Objectives

1. **Efficient Word Prediction**:
Develop an efficient word prediction system capable of suggesting the next word based on the user's input **using n-grams** and **stupid backoff technique.**

2**. Auto-Completion for incomplete word**:
Implement an auto-completion feature that suggests words, even when the input string ends with an incomplete word using a trie for unigrams.

3. **Probabilistic Language Model:**
Apply the **Markov Assumption** to predict the next word by considering only the most recent one or two words from the input.

Problem Definition

1. The challenge is to build a real-time word prediction system that efficiently predicts the next word based on user input using n-grams.

2. We also need to manage data efficiently with minimal computational overhead.

3. Handle incomplete word inputs through the trie data structure.

4. Build the system with a modular design, allowing for future extensions like higher-order n-grams.

Scope and boundaries

- **Static Dataset**: The system operates on a **static dataset** of n - grams . Any new words or n-grams must be manually added. This limits the system's ability to dynamically adapt.

- **Terminal-Based User Interface**: The project will have a **command-line interface** for user input, and suggestions.

- **Non-Adaptive**: The system does not learn from user behaviour over time; it provides suggestions based on the pre-existing n-gram dataset only.

Methodology

1. **Markov assumption**

Instead of computing the probability of a word given its entire history, we can approximate the history by just the last few words.
The tri-gram model, for example, approximates the probability of a word given all the previous words :

$$P(w_n|w_{1:n-1})$$

by using only the conditional probability of the preceding two words:

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1})$$

2. **Stupid Backoff Technique**

Stupid backoff is a smoothing technique used in language models, such as the trigram model, to handle cases where a higher-order n-gram (e.g., trigram) is not found in the training data. The approach avoids assigning a zero probability to unseen n-grams by falling back to lower-order n-grams, such as bigrams or unigrams.

$$S(w_i|w_{i-N+1:i-1}) = \begin{cases} \dfrac{\text{count}(w_{i-N+1:i})}{\text{count}(w_{i-N+1:i-1})} & \text{if count}(w_{i-N+1:i}) > 0 \\ \lambda S(w_i|w_{i-N+2:i-1}) & \text{otherwise} \end{cases}$$

The backoff terminates in the unigram, which has score S(w) = count(w)/N.
Where N is the total count of unigrams.
Brants et al.(2007) find that a value of 0.4 worked well for λ.

Data Structures

Major data structures :

1. **Trie**: The Trie is used to store unigrams (individual words) along with their frequency counts.

- **Fast Lookups**: The Trie provides O(n) lookup times, making it suitable for a word predictor.
- **Efficient Memory Usage**: By storing common prefixes only once, the Trie reduces memory consumption compared to other data structures that would store the same prefixes separately.
- **Ease of Implementation**: The Trie's straightforward implementation supports the project's goal of providing efficient word completion and prediction.

2. **B+ Trees**: B+ Tree for storing and efficiently retrieving n-grams.

   - **Scalability** : It is advantageous due to its ordered nature and ability to handle a large amount of data in a sorted and balanced way. B+ Trees can grow dynamically to accommodate increasing amounts of data.
   -
   - **Facilitating Backoff Mechanism**: The B+ Tree's ability to efficiently retrieve n-grams supports the stupid backoff mechanism,

Supporting data structures :

3. **Dynamic Arrays**: To manage lists of potential word suggestions during the prediction process.

4. **Priority Queues**: To prioritise word suggestions based on their frequency counts or probabilities.

5. **Stacks and Queues**: For managing intermediate states during traversal, insertion, and deletion operations within the Trie and B+ trees.

Future Enhancements

- **GUI / TextEditor**: The autocomplete suggestions can be integrated in a text editor and can be bubbled at real-time. Context-based suggestions can be integrated using deep learning algorithms.

- **Higher model N-gram**: Currently, the algorithm uses tri-gram to return the word. To improve the suggestions and implement context based words, higher n-grams would be helpful. These would further help in correct word predictions.

- **Parts of Speech Tagging**: POS tagging involves assigning grammatical tags (like noun, verb, adjective, etc.) to each word. This information can be used to improve context-awareness .

References

1.Jurafsky, D., & Martin, J. H. (2024). N-gram language models. In Speech and Language Processing (3rd ed.). Draft of August 20, 2024.

2. Efficient in-memory data structures for n-grams
Indexing .Daniel Robenek, Jan Platoš, Václav Snášel
Department of Computer Science, FEI, VSB – Technical University of Ostrava

3.Natural Language Processing with Probabilistic Models
by DeepLearning.AI. ***Taught by:Younes Bensouda Mourri*** *and* ***Łukasz Kaiser****,*