Systematic Analysis of Kaggle Problem: Santa 2024

Barrera Mosquera Jairo Arturo 20222020142
Martinez Silva Gabriela 20231020205

March 2025

1 Problem Overview

1.1 Competition Goal

The goal of this competition is to minimize the perplexity of scrambled text passages by rearranging (permuting) words into their most coherent and natural order. Participants must find the optimal permutation for each passage, ensuring that all original words are used without modification. The evaluation metric is the average perplexity, calculated using the Gemma 2 9B language model—where a lower perplexity score indicates a more fluent and meaningful sentence structure.

1.2 The Dataset

The dataset consists of scrambled text passages derived from classic Christmas stories. It includes two columns:

- **id:** A unique identifier for each scrambled passage.
- **text:** A scrambled sequence of words from a Christmas-themed passage. Each row represents a different sequence.

1.3 Constraints

Words can only be reordered; they cannot be modified, removed, or added.

1.4 The Data

The competition provides scrambled text passages that must be reordered to minimize perplexity.

- **Training Data:** Kaggle provides a sleigh-full of jumbled text passages, each one based on a beloved Christmas story. A metric is included to guide

participants—indicating how puzzled a reader would be by the mixed-up words.

- **Test Data:** Submissions must consist of word permutations given in the `sample_submission.csv` file. Perplexity is computed using the Gemma 2 9B model. The final score is the average perplexity across all sequences.

1.5 The Submission

For each `id`, the submission file must contain a sequence of words in the `text` field. The sequence must be a valid permutation of the original base sequence from `sample_submission.csv`. For example, if the base sequence is: `a b c`, valid submissions include: `a b c`, `a c b`, `b a c`, `b c a`, `c a b`, or `c b a`.

2 System Analysis Report

To reorder scrambled text into meaningful sequences, an autoregressive language model must evaluate the fluency and likelihood of different word arrangements. This is achieved by leveraging next-token prediction capabilities, which assess the natural flow of language based on learned contextual patterns. A pre-trained model such as Gemma 2B, optimized for generative tasks, can compute perplexity scores to identify the most coherent permutations.

2.1 Elements

2.1.1 Input (The Data)

Input Structure: The input consists of multiple scrambled words that form nonsensical sequences. Each input row is a unique set of words that must be reordered.

Input Relevance: The system is entirely dependent on the given input data; it is essential for the model's operation and performance.

Input Processing: Due to the nature of permutations, only one sequence can be evaluated per iteration. Predicting all sequences simultaneously is computationally intensive. The transformation from a scrambled sequence to a coherent sentence follows the data flow shown below:

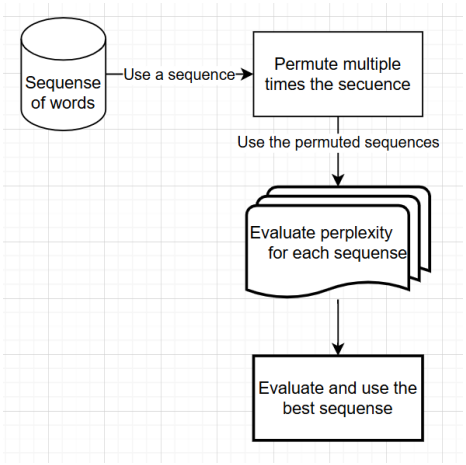

Figure 1: System Data Flow

Functional Requirements (FRs): The functional requirements related to input and data usage are as follows:

- **Sentence Usage:** The system must operate only on the shuffled sentences provided by Kaggle.
- **Word Constraints:** The system must not add, modify, or remove any words from the original input.
- **Sequence Length Handling:** The system must be capable of processing sequences of varying lengths.

2.1.2 The model

The model structure: The implementation consists of two stages. The first involves selecting different sequences from the possible permutations using Python's `itertools` library. Once a set of permutations is generated, each one is evaluated using Google's large language model, Gemma. The permutation with the lowest perplexity is selected as the final output. Feedback is integrated in the evaluation stage: better permutations produce better perplexity scores.

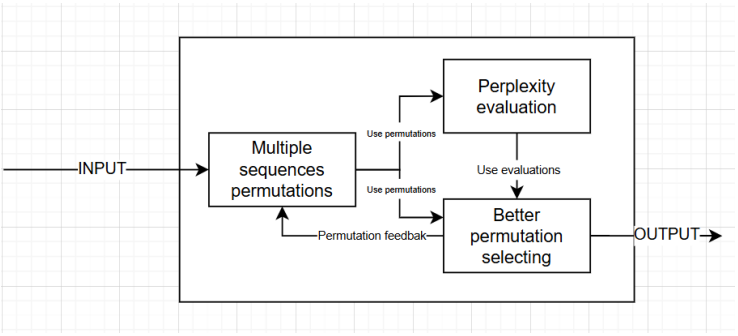

Figure 2: Systems structure

Non-Functional Requirements (NFRs) The non-functional requirements are related to the efficiency of permutation generation and evaluation.

- **Perplexity:** The final sentence should have a low perplexity score.
- **Semantic Coherence:** The resulting sentence must be grammatically and semantically correct.

2.1.3 Output

The output structure: The final submission must contain each predicted sequence paired with its corresponding `id` in a `.csv` file. The relationship is established by the model, which selects the permutation with the lowest perplexity, ideally producing a meaningful and grammatically correct sentence.

2.2 Complexity and Sensivity

2.2.1 Constraints(Restrictions)

There are two major types of constraints:

- **Competition Rules:** Words must be used exactly as provided—no modification, addition, or deletion is allowed.
- **Computational Constraints:** Due to the factorial growth of permutations, evaluating all possibilities becomes computationally expensive as sequence length increases.

2.2.2 Sensitivity

The model is highly sensitive to input permutations. A small change in word order can significantly affect perplexity scores, other factors affecting system senvity:

- **Text Length:** The greater the number of words in a sentence, the more possible permutations exist, making optimization more difficult.

- **Grammatical Structure:** Some passages may contain ambiguities that impact the perplexity evaluation.
- **Lexical distribution:** Phrases with common words can generate more plausible permutations, while phrases with rare words may have fewer viable options.

2.3 Chaos and Randommess

- **Non-linearity and Chaos:** Small changes in the position of a word can cause a drastic change in perplexity. In some cases, a grammatically incorrect phrase may receive a better score than a correct one.
- **Combinatorial Explosion:** For a sentence with n words, there are $n!$ possible permutations. Evaluating all these combinations is computationally infeasible for high values of n.

3 Conclusions

The Santa 2024 Kaggle competition presents a complex yet intellectually engaging challenge that combines combinatorial optimization with natural language understanding. The task of reordering scrambled word sequences into coherent and meaningful sentences requires not only linguistic knowledge, but also algorithmic strategy and computational efficiency.

By leveraging a large language model such as Gemma 2B, the system can effectively evaluate the fluency of different permutations through perplexity scoring, identifying the arrangement that most closely resembles natural human language.

However, several challenges remain. The factorial growth of possible permutations leads to significant computational overhead, particularly for longer sequences. Additionally, the use of perplexity as an evaluation metric, while generally effective, is not foolproof—grammatically incorrect or semantically ambiguous sequences may occasionally yield low perplexity scores, potentially compromising the quality of the output.

This project offers valuable insights into the intersection of language modeling, algorithmic search, and system design. It highlights the importance of balancing linguistic fluency with computational tractability and lays a solid foundation for future research in structured text generation, model evaluation, and intelligent language-based systems.