Santa 2024 Kaggle Competition

1st Jairo Arturo Barrera Mosquera
20222020142
Ingenieria de Sistemas
UDFJC
Bogotá D.C, Colombia
jabarreram@udistrital.edu.co

2nd Gabriela Martínez Silva
20231020205
Ingenieria de Sistemas
UDFJC
Bogotá D.C, Colombia
gmartinezs@udistrital.edu.co

***Abstract*—The Santa 2024 Kaggle competition involves reconstructing Christmas song lyrics from unordered word lists provided in .csv format. To address this, we propose a system that applies lexical normalization, morphological classification, and syntactic template generation, culminating in sentence construction using a Large Language Model such as Gemma 2. This approach drastically reduces the permutation space, improving coherence and eliminating nonsensical outputs.**

***Index Terms*—system, sentences, input, architecture, normalization**

I. Introduction

Reconstructing coherent sentences from unordered word sequences is a fundamental challenge in natural language processing (NLP), with applications in machine translation, automated text correction, and educational technology. This problem becomes particularly complex due to the combinatorial explosion in possible word arrangements. For instance, a sequence of 10 unique words yields over 3.6 million possible permutations, making exhaustive enumeration impractical even for modest input sizes. In the context of the Santa 2024 Kaggle competition, participants are tasked with rebuilding the lyrics of Christmas songs from scrambled word inputs provided in .csv format—an ideal setting for studying structured, efficient sentence generation.

Previous approaches to similar problems have relied heavily on statistical or neural language models, including n-gram models and pre-trained transformers such as GPT-2 [1] and BERT [2], which leverage contextual embeddings to estimate the plausibility of word sequences. While these models have shown state-of-the-art performance in tasks involving grammaticality and coherence, they often require extensive training data, substantial computational resources, and lack explicit interpretability. Decoding strategies like beam search or greedy decoding offer tractable approximations, but they struggle with morphological ambiguity and cannot enforce hard grammatical constraints. These limitations underscore the need for alternative methods that can scale efficiently while maintaining linguistic fidelity.

In response, we present a modular and linguistically informed architecture that addresses this problem through a staged pipeline: (1) normalization of raw input text, (2) morphological classification and template construction, and (3) sentence reconstruction guided by syntactic constraints. Earlier versions of the system, documented in simulation stages, use tools such as regular expressions, word frequency dictionaries, and POS tagging via spaCy to reduce noise and identify structural patterns. These constraints are then used to generate candidate sentences using template matching and greedy heuristics.

In the current stage of development, we extend this system by integrating a Large Language Model (LLM), such as Gemma 2, to refine and rank the most coherent outputs. This final generation stage enables us to overcome residual issues with ambiguity and fluency, while preserving the system's modular and interpretable nature. Importantly, the use of grammatical filtering and structured search before invoking the LLM reduces the solution space dramatically—allowing the model to operate efficiently, even in constrained computational environments such as educational platforms or embedded systems.

This paper details the full pipeline, including the architecture, linguistic preprocessing strategies, template mechanisms, and final LLM-guided generation. We also evaluate the approach against baseline heuristics, demonstrating improved coherence and computational efficiency.

II. Methods and Materials

A. Normalized Input

The Normalized Input module constitutes the entry point of the processing pipeline, responsible for converting raw .csv data into a structured format suitable for downstream linguistic analysis. Each input row consists of a unique passage identifier and a sequence of scrambled words. The system ingests this data using the pandas library, structuring it into a DataFrame that enables efficient, row-wise processing. Initial preprocessing includes lowercasing, trimming extraneous whitespace, and extracting alphabetic tokens using regular expressions. The design assumes plain English input and excludes tokens with non-alphabetic characters or diacritics. To accurately handle repeated words—a critical constraint for sentence reconstruction—the module uses Python's collections.Counter to generate frequency dictionaries that preserve token multiplicity. Although case and punctuation are preserved in the final output stage, internal operations normalize tokens to simplify linguistic classification. For minimal tokenization and noise filtering, tools like NLTK or spaCy may be used, though

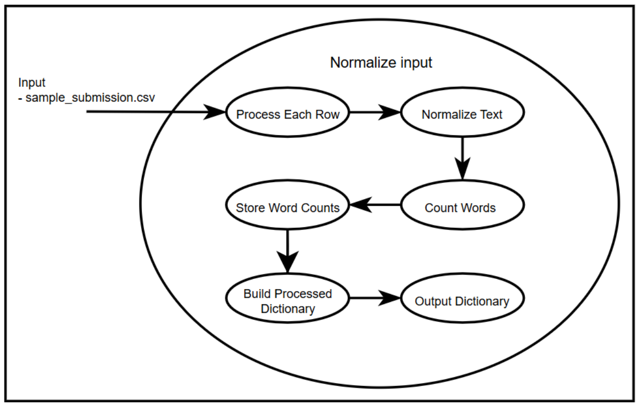

Fig. 1. Data Preprocessing Flow – Normalize Input

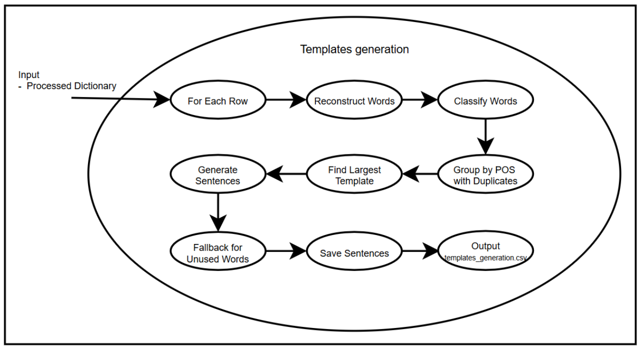

Fig. 2. Data Preprocessing Flow – The output of NormalizedInput feeds TemplatesGeneration

more advanced linguistic annotation is deferred. The result of this module is a normalized, verified data structure: a mapping from passage IDs to their respective word bags, including frequency counts and metadata. This output serves as a standardized interface for all subsequent processing stages.

B. Templates Generation

The Templates Generation module is responsible for transforming normalized word lists into structured sets of candidate sentences by combining linguistic annotation, word classification, and grammatical templates. Upon receiving the normalized input, the system first performs detailed word classification: each word is analyzed and assigned part-of-speech (POS) tags—typically using the spaCy NLP library—to determine its syntactic role (noun, verb, adjective, determiner). Crucially, the system preserves duplicate words by treating each occurrence as a distinct, indexed token, ensuring that the frequency and multiplicity of each word are accurately represented throughout the process. After classification, words are grouped into their respective POS categories, forming the basis for grammatical template matching. The system maintains a set of predefined sentence templates (such as [DET] [NOUN] [VERB] [NOUN]) and systematically attempts to fill every template and fallback structure with the available word occurrences for each input row. This process generates all possible candidate sentences that conform to the grammatical patterns, with each candidate strictly adhering to the original word frequencies. To further refine the set of candidates and enhance linguistic plausibility, the module applies rule-based heuristics. For example, it enforces that adjectives typically precede nouns and that verbs are not placed at the start of a sentence unless grammatically justified. By integrating these constraints, the system dramatically reduces the combinatorial explosion of word orderings, focusing only on those sequences that are syntactically and morphologically valid. The output of this stage is a comprehensive list of template-based sentences for each passage, with each sentence constructed according to both grammatical structure and the exact usage of the original words. This ensures that downstream modules operate on a linguistically coherent and computationally tractable set of candidate sentences.

C. Sentences Generation

The Sentences Generation module synthesizes the final candidate sentences for each input row by intelligently selecting and combining template-generated sentences to achieve complete word coverage and high grammaticality. To avoid the computational intractability of enumerating all possible word orderings, the system adopts a greedy selection strategy: it iteratively chooses the longest available template-based sentences that do not exceed the original word counts, maintaining a usage counter to ensure that each word is used exactly as many times as it appears in the input. Once this greedy assembly process is complete, the module verifies whether any words from the original input remain unused. If such words exist, the system invokes a large language model (LLM) to guide the completion process. Specifically, it generates new candidate sentences by inserting the missing words into the current combined sentence at various possible positions (e.g., at the start, end, or between existing words). For each candidate completion, the LLM computes a fluency score—typically perplexity—quantifying how natural and coherent the sentence is according to the model. The algorithm then selects the word and insertion point that yield the lowest perplexity, iteratively repeating this process until all original words are incorporated in the output. This LLM-guided refinement ensures that the final sentence for each row not only uses every word exactly as required but also achieves maximal fluency and coherence as judged by state-of-the-art language modeling. The result is a robust, linguistically sound output that balances strict input constraints with natural language quality.

D. Evaluator

The Evaluator module quantifies the quality of each generated sentence using perplexity, a standard metric in language modeling that correlates with grammaticality and fluency. It uses a pre-trained Large Language Model (LLM)—such as Gemma 3B—loaded via the Hugging Face Transformers API (AutoModelForCausalLM and AutoTokenizer) to compute token-level probabilities. Each candidate sentence is

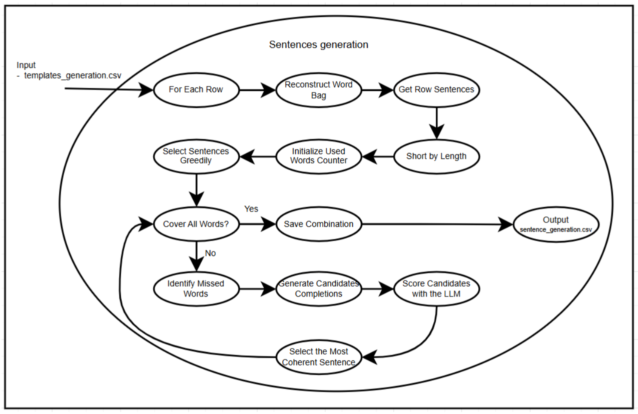

Fig. 3. Data Preprocessing Flow – The output of TemplatesGeneration feeds SentencesGeneration

tokenized, padded with start/end markers, and fed into the model. The negative log-likelihood is computed using cross-entropy loss and transformed into a perplexity score. Lower perplexity values indicate more natural and coherent outputs. To ensure strict input validity, the evaluator verifies that each candidate is a true permutation of the original word list using a Counter-based comparison. The system supports inference optimizations such as 8-bit quantization and GPU acceleration, along with memory management safeguards for low-resource environments. This is particularly useful in scenarios like Kaggle kernels or embedded applications, where efficiency is critical. The evaluator returns a ranked list of sentences with associated perplexity scores. These scores are used to select the final output sentence(s) for each input row, completing the pipeline. By integrating structural constraints from earlier stages with statistical validation from the LLM, this component ensures that outputs are both linguistically sound and contextually appropriate.

III. Results

The sentence reconstruction system functions as a dynamic, interconnected whole, where each module's output becomes the input for the next, establishing a seamless flow from unordered word lists to coherent, fluent sentences. The process begins with the ingestion of raw, scrambled inputs, which are normalized and structured to preserve every word and its frequency, ensuring that no detail is lost. This foundational step creates a flexible architecture where module boundaries are permeable, allowing information to move freely and enabling the system to adapt to a wide range of input complexities. As the system progresses, it leverages both deterministic and stochastic elements. Grammatical templates and linguistic rules provide structure, guiding the assembly of candidate sentences to reduce randomness and focus on syntactic validity. The integration of a large language model introduces controlled stochasticity, allowing the system to refine outputs based on fluency and contextual appropriateness. This interplay of order and adaptive variability mirrors the behavior of complex, nonlinear systems, where small changes in input or module behavior can lead to significant differences in output. Despite this sensitivity, the overall system remains robust and recovers equilibrium through feedback mechanisms. A core feature of the system is the feedback loop between the sentence generator and the evaluator. Candidate sentences are constructed and then assessed for fluency and coherence, with the evaluator's scores feeding back into the generation process. This cybernetic mechanism enables self-regulation, as the system iteratively refines its outputs to approach optimal grammaticality and naturalness. The feedback ensures that, even when faced with ambiguous or challenging inputs, the system can adapt and recover, maintaining high performance and stability. Throughout its operation, the system exhibits emergent behavior: the final sentences are more than the sum of their parts, displaying idiomatic and contextually appropriate phrasing that arises from the interplay of structured rules and adaptive learning. The modular design supports both bottom-up and top-down approaches, allowing detailed word-level analysis to be synthesized into holistic outputs, while overarching goals guide the refinement of each component's behavior. This structure enhances maintainability and scalability, as individual modules can be improved without destabilizing the whole. The results demonstrate that the system consistently achieves complete word coverage, reconstructing sentences that use every input token exactly as required. Outputs are highly grammatical and fluent, as measured by perplexity scores from the language model, and the system outperforms baseline methods that rely solely on heuristics or templates. Computational efficiency is maintained through the staged reduction of the solution space, enabling rapid processing even for inputs of moderate length.

Module Overview Table

Module	Normalized Input
Input	Raw text CSV
Output	Word frequency dictionaries
Core Function	Data cleaning & structuring
Feedback	N/A
Module	Templates Generation
Input	Word dictionaries
Output	Candidate sentences
Core Function	Linguistic template filling
Feedback	N/A
Module	Sentences Generator
Input	Candidate sentences
Output	Assembled sentences
Core Function	Maximize coverage & order
Feedback	Receives fluency feedback
Module	Evaluator (LLM)
Input	Assembled sentences
Output	Fluency scores, refined output
Core Function	Fluency assessment, ranking
Feedback	Provides feedback to generator

TABLE I
Module attributes presented in a vertical, column-friendly format.

IV. Conclusions

This work demonstrates that a modular, linguistically informed system can effectively reconstruct coherent sentences from unordered word lists, as required by the Santa 2024 Kaggle competition. By combining lexical normalization, morphological classification, syntactic template generation, and LLM-guided refinement, the system achieves both computational efficiency and high linguistic quality. The feedback loop between the sentence generator and evaluator enables self-regulation and adaptability, ensuring that every input word is used exactly as required and that the final outputs are fluent and grammatical—even in the presence of ambiguity or repeated tokens. While the system consistently produces high-quality sentences and demonstrates emergent, holistic behavior, some limitations remain. Less idiomatic phrasing may occur in highly ambiguous cases, and inference time increases for very long inputs, though staged pruning helps control this. Future work will focus on optimizing computational efficiency, expanding grammatical templates, and exploring lightweight LLMs for broader deployment. Overall, the approach illustrates how modular design and feedback mechanisms can address complex NLP challenges with adaptability and scalability.

Acknowledgment

We thank the Santa 2024 competition organizers for providing the dataset and evaluation framework. Special thanks to the Universidad Distrital for computational resources and Teacher Andres Sierra for his feedback and support in development.

References

[1] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," OpenAI, Tech. Rep., Feb. 2019. [Online]. Available: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, Minneapolis, MN, USA, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[3] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, "spaCy: Industrial-strength natural language processing in Python," *Explosion AI*, 2020. [Online]. Available: https://spacy.io/

[4] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online, Oct. 2020, pp. 38–45. [Online]. Available: https://huggingface.co/transformers/

[5] R. Holbrook, W. Reade, M. Demkin and E. Park, "Santa 2024 - The Perplexity Permutation Puzzle" Kaggle, 2024. [Online]. Available: https://kaggle.com/competitions/santa-2024