Text Normalization and Composition Writer Information Prediction: A Hybrid Approach

---

Overview

This report describes the full implementation of the Text Normalization Assessment, providing a high-level overview of the approach used. The project leverages a hybrid method that combines rule-based models with Large Language Models (LLMs) to improve text normalization, particularly for ambiguous or non-English text. This approach enhances the accuracy and quality of text processing, ensuring more reliable and effective results for real-world applications.

General Key Objectives

- **Normalize Text**: Correct inconsistent or ambiguous text to improve prediction accuracy, ensuring cleaner and more reliable data.
- **Handle Non-English Text**: Detect and process non-English characters and edge cases effectively, enhancing the model's ability to handle diverse languages and scripts.
- **Hybrid Models**: Combine various approaches (rule-based and machine learning) to enhance the reliability and accuracy of predictions, ensuring more robust performance across different cases.
- **Data-Driven Evaluation**: Evaluate the variety of approaches using multiple metrics, including Exact Match Percentage, Word Error Rate (WER), and Character Error Rate (CER), to ensure decisions are driven by performance data.

- **Tracking Modifications**: Monitor the extent to which LLMs modify or add words during text processing. The primary objective is to focus on text normalization by removing inconsistencies, without introducing unnecessary changes that could impact the integrity of the data.
- **Subset Experimentation**: Conduct experiments using a subset of the dataset, allowing for faster experimentation, optimization of models, and tuning of execution times. This subset provides a controlled environment for testing different approaches and refining models before scaling to larger datasets.

Exploratory Data Analysis (EDA)

The goal of the Exploratory Data Analysis (EDA) was to understand the dataset's structure and inform our approach to preprocessing and model development. Key findings and action items include:

Key Findings:

- **Missing Data**: Approximately 1,400 missing values in the Ground Truth column, which requires imputation or handling to ensure dataset integrity.
- **Text Length**: The text length distribution was analyzed, revealing the average length, which will inform decisions on text normalization to handle varying text sizes effectively.
- **Language Distribution**: A significant mix of Latin and non-Latin text was identified, which influences preprocessing choices. Proper handling of diverse languages and characters is critical for accurate normalization.
- **N-Gram Patterns**: Bigram, trigram, and 4-gram analysis revealed frequently occurring word pairs and sequences, which will be used to enhance feature engineering and improve model predictions by capturing context from text sequences.
- **Special Characters**: Special characters and punctuation were found to be prevalent in the Original_Text, guiding the development of text

cleaning processes to remove unnecessary symbols and standardize the text.

- **Unique Word Count**: A diverse vocabulary was observed in the text, indicating the need for robust tokenization and vocabulary management techniques to ensure effective processing of varied words.
- **Handling Specific Symbols**: A function was implemented to identify and sample cases where specific symbols or sequences, like `<Unknown>`, appear in the `Original_Text` column. This is useful for detecting problematic patterns in the data that might require special handling or cleaning.

Action Items:

- **Handle Missing Data:** Missing values in the Ground Truth column are handled by converting np.nan values to a string format, ensuring that no data is lost and the dataset remains consistent for further processing.
- **Text Normalization:** The text normalization process includes the removal of special characters, proper handling of non-Latin text, and ensuring consistent formatting across all entries. This step ensures that text input is standardized and ready for analysis.
- **Feature Engineering:** We leveraged N-grams (bigrams, trigrams, and 4-grams) to capture meaningful text patterns, which will enhance the model's ability to understand the structure and relationships within the data.
- **Text Cleaning:** Punctuation and symbols were standardized to reduce noise in the text data. This cleaning step ensures that extraneous characters do not interfere with the quality of the input.
- **Optimize Tokenization:** The tokenization process was optimized to effectively handle a diverse vocabulary, improving the model’s ability to generalize across various text entries.
- **Text Length Distribution & Latin Character Rows:** The analysis of the mean text length distribution provided insights into how the text varies in size. Additionally, we identified the number of rows

containing Latin characters, which helped inform the preprocessing strategy, especially for non-Latin text handling.

Approaches Implemented

1. **Baseline Rules**
   Rule-based text normalization techniques using a set of predefined rules. These rules address basic tasks, such as removing common stopwords and simplifying inconsistent text.
   Challenges: This approach struggles with ambiguous or complex cases, particularly when dealing with non-English characters, making it less flexible and efficient for diverse datasets.

2. **Enhanced Rules**
   Building upon the baseline rules, the Enhanced Rules approach incorporates an enriched set of stopwords and additional logic to handle more complex edge cases. This involves rules to better deal with non-standard formatting, such as multiple spaces, inconsistent capitalization, and special characters. The enhanced set of rules aims to improve the accuracy of text normalization.
   Challenges: While more flexible, this approach is still limited compared to machine learning models, as it lacks the capability to adapt dynamically to new or unseen patterns in the data.

3. **LLM-Assisted Approach**
   This approach leverages Large Language Models (LLMs) to refine text based on a broader contextual understanding. The LLM is instructed using specific rules and a few-shot learning approach to help it better comprehend and normalize ambiguous text. By providing examples of common text patterns and normalization tasks, the LLM becomes more capable of handling complex text, especially with non-English characters or irregular formatting.
   Challenges: LLM-assisted models are computationally expensive,

and the processing time can become a bottleneck when dealing with large datasets.

4. **Voting Hybrid Approach**
   A hybrid approach that combines the outputs of multiple models—Baseline, Enhanced, and LLM—using a voting mechanism to decide the final prediction. The final decision is based on the majority vote of the individual model predictions. This allows the model to combine the strengths of each approach, balancing simplicity and flexibility.
   Challenges: The main challenge here is the potential for noise if the predictions from the different models vary significantly. The voting mechanism must manage such discrepancies to maintain prediction reliability.

5. **RefineBoost Hybrid Approach**
   This approach combines rule-based models with boosting techniques to dynamically refine predictions. After the Enhanced Rules model generates an initial prediction, a new LLM is instructed to refine the output if necessary, improving overall accuracy. The boosting mechanism ensures that errors from the initial prediction are minimized, leveraging the LLM's ability to correct mistakes.
   Challenges: This approach is more computationally demanding compared to simpler models, as it requires multiple layers of processing to achieve the best result.

6. **Conditional Hybrid Approach**
   The Conditional Hybrid approach selects the best model depending on the characteristics of the text. For example, when dealing with non-English characters or texts with large lengths, the LLM is used to handle the complexity and diversity. In other cases, the simpler Baseline model is preferred for its speed. This dynamic approach allows for the best model to be applied based on the specific text characteristics, ensuring a balance between accuracy and efficiency.
   Challenges: This approach is more time-consuming because it

involves additional logic to determine when to switch between models. However, it offers a better balance of accuracy and efficiency, making it ideal for handling diverse datasets.

Metrics Evaluated

1. **Exact Match Percentage**:
   Measures the percentage of exact matches between predicted and true values. This metric gives a clear indication of how well the predicted text matches the ground truth in terms of accuracy.
2. **Jaro-Winkler Similarity**:
   Measures the similarity between two strings by considering the number of matching characters and their proximity. This metric is particularly useful for detecting minor spelling errors or slight variations in the text.
3. **Word Error Rate (WER)**:
   Measures errors at the word level. It calculates the difference between the predicted and actual words, considering insertions, deletions, and substitutions.
4. **Character Error Rate (CER)**:
   Measures errors at the character level, similar to WER but at a finer granularity. It highlights character-level discrepancies between the predicted and actual text.
5. **Execution Time**:
   Measures the time taken to process the dataset for each approach. This metric is crucial for assessing the efficiency of each method, especially when working with large datasets.

Results and Findings

We conducted experiments using a subset of 100 rows from the dataset, a decision driven by time and resource constraints, particularly the computational demands of Large Language Models (LLMs). This smaller

sample size allowed us to efficiently run multiple experiments and compare the performance of different approaches on the same subset. By working with this controlled batch, we were able to rapidly iterate, fine-tune the models, and assess the effectiveness of various methods in a more time-efficient manner, while ensuring consistency in evaluation across all approaches.

Approach	***Exact Match (%)***	***Jaro-Winkler Similarity***	***WER***	***CER***	***Time (mins)***
Baseline Original	63.00%	0.85	0.39	0.10	0.00
Baseline Rules	65.00%	0.86	0.38	0.07	0.01
Enhanced Rules	69.00%	0.86	0.20	0.06	0.00
LLM-Assisted	75.00%	0.93	0.25	0.07	11.86
Voting_Hybrid	70.00%	0.87	0.25	0.08	0.00
RefineBoost_Hybrid	68.00%	0.86	0.21	0.07	2.13
Conditional_Hybrid	75.00%	0.93	0.24	0.07	11.94

Comments on the Results Table

- The **Conditional Hybrid** approach achieved the highest **Exact Match** score (75%) and showed strong performance in terms of **Jaro-Winkler Similarity** (0.93), **WER** (0.24), and **CER** (0.07), making it a well-rounded solution for text normalization. However, it comes with the highest execution time (11.94 minutes), highlighting the trade-off between accuracy and speed.
- **LLM-Assisted** models also excelled in terms of accuracy, achieving the second-highest **Exact Match** score (75%) and performing well in reducing **WER** (0.25) and **CER** (0.07). Despite these strengths, the execution time (11.86 minutes) was relatively long due to the computational demands of the LLMs.
- **Voting Hybrid** performed decently with an **Exact Match** of 70% and **Jaro-Winkler Similarity** of 0.87, showing strong results for word-level accuracy. It also had one of the lowest execution times (0

minutes), indicating that it can be a good option when speed is a priority, though with a slight drop in accuracy compared to LLM-assisted models.
- **RefineBoost Hybrid** performed closely to **Enhanced Rules**, showing a good balance of **WER** (0.21) and **CER** (0.07) with **Exact Match** at 68%. This approach showed a moderate execution time (2.13 minutes), making it a suitable candidate for scenarios requiring better performance without extreme computational costs.
- **Enhanced Rules** achieved a reasonable **Exact Match** score of 69% and exhibited low error rates (**WER** 0.20, **CER** 0.06), while maintaining a fast execution time (0 minutes). This model is more computationally efficient and offers a good balance for tasks with moderate accuracy requirements.
- **Baseline Rules** performed the fastest, but with the lowest **Exact Match** (63%), higher **WER** (0.39), and **CER** (0.10). While it is efficient, it struggles with more complex cases, particularly when handling ambiguous or non-English characters.

Insights

- **Hybrid Models Perform Best:** The **Conditional Hybrid** approach stands out, offering the highest **Exact Match** while balancing accuracy with efficiency. It is ideal for scenarios where high precision is crucial, though it comes with higher computational costs.
- **LLM-Based Approaches Reduce Errors:** Both **LLM-Assisted** and **Conditional Hybrid** models significantly reduced **WER** and **CER**, demonstrating their ability to handle complex text normalization tasks, despite their longer execution times.
- **Rule-Based Models:** While **Enhanced Rules** provided a solid compromise between speed and accuracy, **Baseline Rules** performed poorly in comparison, especially in terms of text normalization quality.

Performance Trade-offs

- **Latency vs. Accuracy:** Moving from **Baseline Rules** to more advanced models like **LLM-Assisted** and **Conditional Hybrid** yields a substantial improvement in accuracy. However, this comes with a clear trade-off in processing time, as the more accurate models take longer to run.
- **Cost vs. Performance: LLM-Assisted** and **Conditional Hybrid** models offer the best accuracy but are computationally expensive. For environments where resources are constrained, models like **Voting Hybrid** or **Enhanced Rules** present better trade-offs, offering good performance at a fraction of the computational cost.

Future Work

1. **Experimentation with Heavier LLMs**
   We plan to explore more powerful LLMs such as **GPT-4**, **T5**, and **PaLM** to push the boundaries of accuracy, particularly for tasks that require deeper contextual understanding and nuanced processing. These models are expected to enhance our ability to handle complex and ambiguous cases, providing better generalization across diverse text types.
2. **More Sophisticated Hybrid Approaches**
   To further improve the performance of our models, we will investigate advanced hybrid strategies:
   - **Ensemble Learning:** By combining multiple models through techniques like stacking, bagging, and boosting, we aim to improve robustness, reduce overfitting, and leverage the strengths of different approaches.
   - **Fine-Tuned Models for Specific Tasks:** We will fine-tune more powerful LLMs on domain-specific datasets, optimizing them for specific tasks to increase task-specific accuracy and efficiency.
3. **Integration of Pretrained Models**
   We will experiment with **pretrained models** for specialized tasks

such as **Named Entity Recognition (NER)** or **language translation**. Integrating these models will provide a significant boost in accuracy for these specific applications, leveraging the knowledge embedded in these pretrained systems.

4. **Further Prompt Engineering and Temperature Tuning**
   Refining **prompt engineering** and experimenting with **temperature tuning** will enable us to guide LLMs more effectively, encouraging them to generate more precise and contextually relevant outputs. This will be particularly useful for ensuring consistency in text normalization tasks.
5. **Scaling for Larger Datasets**
   As we continue to improve our models, we plan to expand our dataset to test the scalability of our approaches. This will allow us to evaluate the feasibility of deploying these models in real-time, as well as assess their performance and resource consumption on larger-scale datasets.

Conclusion

This project highlights the effectiveness of hybrid models, particularly those that integrate rule-based systems with LLMs, in achieving reliable results for text normalization and composition writer information prediction. Among the approaches tested, the **Conditional Hybrid** method provides the optimal balance between accuracy and execution time, making it particularly suitable for handling complex text tasks with diverse characteristics.

Looking ahead, future work will explore the potential of more advanced LLMs like **GPT-4**, as well as incorporating **ensemble learning** and **fine-tuning** for domain-specific tasks. These strategies will enhance model performance, enabling greater accuracy and more efficient handling of specialized text normalization scenarios. Additionally, scaling our models to larger datasets will be crucial for assessing their real-world applicability, particularly for real-time applications in production environments.

In conclusion, after all the experiments and computational optimizations, we discovered that running a **GPT-4o Mini** model costs a modest **2$**... though we might need to factor in the cost of coffee and computational resources to get the full picture! 😄