Daniera Nanda Ariefti | Explanation of Phyton Test

Machine Learning Test

Given the limitations of my laptop's processor, I worked with a subset of 10,000 data points to maintain manageable processing times and prevent overloading system resources. For this name correction task, I initially applied FuzzyWuzzy, a straightforward string matching library, to assess and standardize company name variations. By calculating token-based similarity scores, FuzzyWuzzy is able to effectively identify minor variations, typos, or slight formatting differences (e.g., "Teaboard Limited" vs. "Teaboard Lmited"). This method is lightweight and performs well for relatively minor name variations.
To explore potential enhancements through machine learning, I also experimented with BERT (Bidirectional Encoder Representations from Transformers), a pre-trained model that generates high-dimensional vector embeddings for each name, capturing contextual and semantic relationships. By comparing these embeddings, BERT can identify conceptually similar names, even when textual matches are not exact. This vector-based similarity search shows promise for managing complex name variations and multilingual data.
In this case, however, both FuzzyWuzzy and BERT achieved similar accuracy, with BERT offering no significant improvement. After careful consideration, I determined that further training of the BERT model was not necessary, given a few factors:

1. Class Overflow
   The dataset contains a high number of unique classes (company names), which could overwhelm the model, making it challenging to generalize to new data.
2. Insufficient Variation in Data
   Limited diversity in name patterns means that complex model training might not yield additional benefits. FuzzyWuzzy's simpler approach proved effective in this context.
3. Risk of Overfitting
   Training a model on this dataset might lead to overfitting, where the model learns specific idiosyncrasies instead of general patterns, potentially resulting in inaccuracies with new data.

Considering these factors, the complexity and computational cost of implementing a machine learning model do not seem justified in this case. FuzzyWuzzy provides a simple, efficient, and effective solution, demonstrating that traditional methods can sometimes perform as well as advanced machine learning models—especially when constrained by data and computational resources.

Adding New Data

To make the yearly addition of new firm data more efficient, an incremental approach could be highly beneficial. By establishing a persistent database to store all cleaned names, IDs, and metadata from previous years, we can streamline the process. When new data arrives, we would clean and standardize only the new entries, then match them against existing entries in the database using vector-based similarity (e.g., BERT embeddings) or fuzzywuzzy. If a match is found, we can reuse the existing ID; if not, a new unique ID would be assigned, and the entry added to the database. Automating this process with scheduled scripts would allow for regular checks and updates, minimizing redundant re-processing and making yearly updates more seamless and scalable.

In more detail, the code optimizes this process by leveraging an existing cleaned database (cleaned_database.csv). It first loads both the database, containing previously cleaned names and IDs, and the new data for the current year (e.g., ForeignNames_2022.csv). The new data is then standardized and cleaned with the clean_firm_name function to ensure consistent formatting. Next, the code performs incremental matching by checking each name in the new data against the existing database, using either fuzzywuzzy or BERT embeddings. If a match is found (with a similarity score ≥ 95), the existing cleaned_ID is reused. If no match is found, the name is treated as new, and a unique ID is generated. This incremental approach allows the code to process only the new entries, conserving time and resources.

Finally, the matched and newly assigned entries are appended to the existing database, and the updated database is saved back to cleaned_database.csv. This approach facilitates annual updates without the need to re-run the entire cleaning and matching process, ensuring both efficiency and scalability.