Comments - Python Test

Jairo Enrique Galvis Avella

Methodologhy

I start by loading and cleaning the firm names data from `ForeignNames_2019_2020.csv` and mapping it with country names from `Country_Name_ISO3.csv`. First, I clean up any messy country names and make sure everything is formatted consistently. Then, I clean the firm names by country, handling variations and stopwords that might differ from one place to another. Using fuzzy matching, I group similar names together and give each firm a unique ID that includes the country code to keep track of where they're from. I also train a machine learning model on a portion of the data (70% for training and 30% for testing) using Naive Bayes with TF-IDF vectorization to predict cleaned names based on the original ones. Moreover, I created two intermediate CSV files to save important datasets obtained in the process because sometimes the kernel crashes or something else happens, and we might lose the data and time (especially when we work with big datasets).

I assume that the input data is generally clean and relevant, with the primary issue being inconsistencies in firm names that need to be addressed. The approach relies on the idea that stopwords and naming conventions might vary from country to country, which justifies the need for country-specific cleaning. I also assume that a fuzzy matching threshold of 80 is effective for grouping similar names without missing important matches or including too many false positives. For the machine learning model, I assume that using Naive Bayes with TF-IDF vectorization is suitable for predicting cleaned names based on their textual features and that the training data is representative enough to help the model generalize well.

There are several trade-offs in this approach. Cleaning names country by country adds complexity and processing time, but it reduces errors and accounts for regional differences in naming conventions. Using a fuzzy matching threshold of 80 strikes a balance between capturing similar names and avoiding excessive false positives, though it might miss some matches or include some errors. The choice of a machine learning model like Naive Bayes with TF-IDF vectorization is effective but assumes that the training data is representative and that the model can generalize well to new data.

Finally, it is important to mention that due to the lack of time, the machine-learning model is not very accurate. In addition, the model only takes a sample of the whole dataset because of its big size. With more time, the model can be improved to predict better the firm names and use the whole dataset to obtain a better normalization process. In this exercise, I tried to put the essential structure according to the task and show the intuition of the model, even though it is still weak to use in real-world practice. Having more time, I would correct as many names for the training dataset to obtain better results. Likewise, I would test other learning models, try different strategies to compare and choose the

best one, and make different tests to guarantee the quality and consistency of the predictions.

Part 2

Now in the real world, this process will be continuously happening, ie, every year you will have to add new firms. How will you optimize your work? Will you run everything again whenever new firms are added or find a way to make frequent updates without rerunning everything? Please include a short description of your update process in a word or PDF document you will name comments_yourfirstname.

In a real-world scenario where you need to add new firms to your dataset each year continuously, optimizing the process to avoid rerunning everything from scratch is essential for efficiency.

Establishing and maintaining a master ID database is crucial for efficiently managing firm names and their associated IDs. This database serves as a persistent storage system that keeps a comprehensive list of all firm names and their unique IDs. Each time new data arrives, you can refer to this master database to determine whether a firm already exists or is new. By doing so, you avoid the need to reassign IDs to firms that have been previously processed. Instead, you can directly retrieve and update records as needed. This approach not only saves time but also ensures consistency across your dataset, as each firm retains its unique ID even as new entries are added.

To further optimize your workflow, adopt an incremental processing approach when incorporating new data. Instead of reprocessing the entire dataset each time, focus only on the new entries. Begin by loading the master ID database to check which firms are already included. Then, compare the incoming data against this database to identify any new firms. Assign unique IDs to these new firms and use the existing IDs for those that are already present. Once this is done, update the master database with the new firms and their IDs. This targeted approach not only reduces the processing time but also minimizes the risk of introducing errors or duplications.

Therefore, using an incremental learning or partial fitting method allows the model to learn from new information without discarding previous knowledge. Regularly evaluate the model's performance to ensure it continues to provide accurate results. By applying this approach, you maintain model efficiency and relevance while adapting to new data trends.

Finally, it is important to incorporate quality assurance practices and safeguard the accuracy and consistency of your dataset, ensuring that the data management process remains effective and dependable.

The folder with the files related to this Python Test is in the following public GitHub repository: https://github.com/jegalvisa/WB-Task.git