Address Normalization

Automating Data Preprocessing

Haider Abbas
Cybernet - TQM

September 2023

Abstract

In today's digitized global environment, the quality of data, especially addresses, plays a pivotal role in business intelligence. For many industries, addresses are paramount data points, where inaccuracies can cause considerable complications. This is particularly true for Pakistani addresses, which, to date, lack a reliable normalization system. This project aims to bridge this gap by introducing a comprehensive address normalization system exclusively designed for Pakistani addresses. Traditional methods, reliant on tools such as Excel, are not only labor-intensive but also prone to human mistakes, leading to disparities. This project seeks to automate this task, converting various address formats into a standardized form based on set guidelines. This automation promises not only time efficiency but also a marked reduction in manual errors. By ensuring uniformity in address data, the project endeavors to amplify data quality, streamline address matching, and refine address-based search and analysis. The project's primary challenges include formulating a universal nomenclature for all Pakistani addresses, grappling with the diverse nature of addresses, extracting building names, and revolutionizing a traditionally manual process to achieve superior accuracy.

Contents

Section 1

Introduction

1.1 About

In a rapidly digitizing world were data has become the new currency, having good quality data is essential for business intelligence. Especially when it comes to addresses, which often serve as critical data points for almost all businesses and services, discrepancies or irregularities can pose significant challenges. This project aims to solve these challenges, particularly in the context of Pakistani addresses where no such solution with a high percentage of success currently exists.

1.2 Objective

The primary objective of this project is to develop a comprehensive address normalization system tailored specifically for Pakistani addresses. Traditionally, address normalization tasks are performed manually on tools like Excel, a process that was not only time-consuming but also prone to human errors, leading to inconsistencies and inaccuracies. By automating this process, our program will transform addresses from diverse formats into a standardized structure that adheres to predefined guidelines and rules. This automated approach not only significantly reduces the time taken but also minimizes errors associated with manual input. By achieving consistency and uniformity in address data, the project aims to enhance data quality, facilitate efficient address matching, and simplify address based searches and analysis.

1.3 Challenges

List of primary challenges:

1. Designing a standardized nomenclature to which all Pakistani addresses can adhere.
2. Heterogeneous nature of addresses.
3. Extraction of building names.
4. Automating a manually done process to achieve greater accuracy.

Section 2

Methodology

2.1 Overview

The following is a high level overview of the entire process:

- Stage 1: Data Preprocessing - string manipulation, standardization & tokenization
- Stage 2: Address Parsing and Normalization
- Stage 3: Building Name Extraction
- Stage 4: Normalization Analysis

2.2 Breakdown

The following provides an in depth explanation of the entire process.

2.2.1 Data Preprocessing

The raw address data undergoes a series of foundational preprocessing steps. These steps are essential to prepare the data for address parsing and normalization stage.

1. Loading the raw data file containing the addresses and dropping irrelevant columns for faster computation. The source file type to be given to the program should be a comma separated values (csv) file.

2. The program includes a unique identifier associated with each address which serves as a primary key. This can be Ticket #, Customer ID, etc. present in the source file.

3. Address is passed through a preprocessing function which does the following to ensure uniformity:

    - Lowercase conversion since our process is case sensitive.
    - Removing multiple commas placed together since comma serves as a delimiter to tokenize the address, multiple commas placed together will give empty tokens.

- Removing punctuation except for "," and "#" is optional although not preferred.
- Standardization of abbreviations. As the addresses are not standardized, there are countless different abbreviation versions of the same word. We have 100+ mappings of abbreviations to standardize the address.
- Removing extra white spaces, ensuring single space amongst characters.
- Identifying address type: House or Apartment. Portions are considered as apartments. Identification is binary and is based on keywords present in the address.
- Tokenizing the address, splitting the address using comma as our delimiter and removing duplicate tokens.

4. At the end of this stage we have the address type, tokenized address and a uniformly formatted address.

2.2.2 Address Parsing and Normalization

This stage focuses on structurally interpreting and breaking down the address into its components. The normalization process then ensures that these extracted components adhere to a uniform standard, allowing for more straightforward comparison, matching and retrieval.

We have designed a uniform address normalization schema applicable for all Pakistani addresses.

Type	House #	Apartment #	Building #	Building Name
Street	Road	Sub Area	Area	City

Table 2.1: Normalization Schema

For any field not existing in an address, there would be None written in that field. For example for an address of type house, there would be no *Building #* and no *Building Name* so there would be None placed there. Similarly if an address does not have a street or if the address is an apartment type but it is a portion so there would be no building details, then there will be a None in the respected fields.

Normalization is being done from the tokenized list based on an elimination based approach where we identify a field and remove that identified token from the tokenized list until we have fully normalized the address.

Following are steps being performed in this stage to normalize the address according to the fields.

1. Placing Address type in the *type* field identified in Stage 1. Example: house or apartment.

2. *City* field has a standard place in the address, which is at the end, i.e the last token. This is true for all addresses. Example: Karachi, Lahore, Multan, etc.

3. *Area* field also has a standard place in the address, which is before the city name, i.e the second last token. This is true for all addresses. Example: Clifton, Defence, PECHS, etc.

4. Many addresses have a common comma placement problem. Since we are using commas to tokenize the address, incorrect or missing commas would result in incorrect tokenization. To resolve this issue we have a field separation checking mechanism in place, which checks based on keywords the current tokenized address if more than one field is present in a single token. If present it will separate the fields based on pattern recognition and place them into different token for each field. Most common field separations are for *House #* & *Street* and *Apartment #* & *Street*.

   Example:
   Before: [{house # abc/123 def street}, {xyz road}, {block a}]
   After: [{house # abc/123}, {def street}, {xyz road}, {block a}]

   Now that we have assured that no token has more than one field, we now start to extract other components one by one.

5. Identifying token(s) for *Road* based on keywords. Example: University road, Khayaban-e-Badar, Shahrah-e-Faisal, Rashid Minhas road, etc.

6. Idenfifying token(s) for *Street* based on keywords. Example: 123rd Street, Lane 1.

7. Idenfifying token(s) for *Apartment #* based on keywords. Example: Apartment/Suite 123, Apartment/Suite 123 4th floor, etc.

8. Idenfifying token(s) for *House #* based on keywords. Example: House # A1, etc.

9. Idenfifying token(s) for *Sub Area* based on keywords. Example: Block 7, Phase 8, Sector 11, etc. This may not ensure complete extraction of Sub Area fields as many do not include the standard keywords.

10. At this point, we now have 3 potential fields left in the tokenized address. For this we have designed a probability based algorithm to classify the remaining fields, aiming to identify the *Building #*, *Building Name* and *Sub Area* based on the sequence of tokens. Below is a high level working mechanism of the function:

    (a) Initial Checks: The function first checks if the tokenized address remaining is empty. If true, the address has already been normalized and proceeds further.

    (b) Score Assignment: The function then assigns a score to each token in the tokenized address. This score, termed as `index_percentage`, represents the token's position in the actual tokenized addresses done in the start as a percentage of the address's total length. The rationale behind this score is to provide a probabilistic measure of the tokens's significance in the overall address.

    (c) Field Classification Based on Score: Fields with an `index_percentage` greater than 50 are considered potential sub areas, while those with a score of 50 or below are regarded as potential building names or numbers. This threshold is based on the fact that fields like sub area are likely to appear later in the address while building number or name are found earlier.

(d) Further Discrimination of Building Name and Number: Among the fields classified as potential building names or numbers, the one with the highest `index_percentage` is considered the potential building name. The rationale is that building numbers tend to precede building names in typical address formats.

(e) Field Extraction: We then classify each token left to it's respective field and place it accordingly.

Another simple way of looking it is such that there are one of two scenario.

If type is house: Up to 1 token left (rarely more than 1 token left). In majority of cases there is no token left and the address is completely normalized, but if there are token(s) left then they belong to the *Sub Area* field are not identified from common keywords identification performed above. Such tokens are also placed in the *Sub Area* field. Example: Bath Island, KDA Market, Near Sultan Masjid, etc.

If type is apartment: Up to 3 tokens left. (rarely more than 3 tokens left) so we will just follow the probability based algorithm defined above to identify field all tokens correctly.

At the end of this we have completely normalized the address and the tokenized list is empty.

11. Placing 'None' were there is no field found.

2.2.3 Building Name Extraction

Building names often pose a significant challenge due to their unique and diverse nature, therefore we have an additional mechanism in place for extracting missing building names. Instead of relying on exact string matching, which can be too strict for real-world applications, the program employs a fuzzy matching technique. This method gauges the similarity between two strings by determining the minimum number of edits (insertions, deletions, or substitutions) required to transform one string into another, which is known as the Levenshtein distance. By doing so, the system can match building names even if they are slightly misspelled or presented differently, ensuring higher accuracy in extraction. To balance between precision and recall, the function incorporates a threshold parameter. This value, ranging between 0 and 100, represents the minimum similarity score for a match. Adjusting this threshold enables users to manage the leniency of the match, catering to specific requirements of the project. We are currently using a threshold of 70, i.e. minimum 70% similarity.

In this stage, we are taking into account those addresses whose type is apartment but there is a None *Building Name* and those addresses which have incorrect building name found which is deduced via pattern recognition. For these addresses, we extract the building name directly from the address string through the above described method with the help of a Building Name corpus.

2.2.4 Normalization Analysis

The Address Normalization Analysis stage is a pivotal checkpoint in the address normalization pipeline. This stage is dedicated to evaluating the accuracy of the work done. The analysis stats will be displayed on the terminal screen where the program is run from (more details in Section 3). Highlights of the analysis is as follows:

- Missing Field Analysis: Computes the number of missing values in specific columns (e.g., 'House #', 'Apartment #', 'Building Name') for distinct types like houses or apartments.
- Type Analysis: Assesses the distribution of address types (e.g., houses vs. apartments) within the data.
- Accuracy Computation: Calculates the overall success percentage of the normalization process based on correctly normalized addresses.
- Detailed Field-Wise Reporting: For every significant address field, it will display out the total count, the number of missing or incorrect entries, and the success percentage.

2.3 Development Tools:

The entire project is completely done in Python (version: 3.11).

Following are the list of essential Python libraries used:

- pandas
- fuzzywuzzy
- nltk
- string
- re

2.4 Alternate Methodologies

Our chosen methodology is the best possible approach for solving this problem (for results refer to Section 4). For validating this statement, it's essential to acknowledge alternative approaches and inquiring why they were not the best fit to our problem. Exploring various strategies provides a comprehensive view of the problem space and offers validation to our approach. This section elaborates on some alternative methodologies and why they are not the best choice.

2.4.1 Machine Learning

Supervised machine learning models can be trained on labeled datasets. One such model already exists for Islamabad based addresses which can be found here, but it achieves only an 83% accuracy rate. While this might seem respectable, it falls short when considering the importance of address accuracy in practical applications along side the fact that Islamabad has the most uniform address as compared to any other city in Pakistan. Moreover, relying solely on a model trained on Islamabad addresses means disregarding the vast regional variations present in Pakistani addresses. If for a model trained on the best possible training data gives an accuracy percentage of 83%, using such an approach for all Pakistani addresses would give much lower accuracy percentage, let alone the computational costs of training a machine learning model. In our manual address normalization process, we achieved an accuracy rate of approximately 80%. Considering this, leveraging machine learning to achieve a similar accuracy may not provide significant added value. We explored the option of training the Google BART model. However, to effectively implement this, it necessitates high-quality training data and advanced computational resources to handle the intensive calculations.

2.4.2 Geocoding APIs

Relying on third-party geocoding services like Google Maps or Here Location Services can seem like a potential approach, given their widespread use and extensive databases. However, when it comes to Pakistani addresses, several challenges present themselves. Firstly, while these services might have comprehensive data for metropolitan areas like Karachi, Lahore, or Islamabad, their databases may lack granularity or be outdated for peripheral or rural regions of Pakistan. Secondly, geocoding services often adhere to international standards for address representation. Pakistani addresses do not align with these standards. Last but not the least, continuous reliance on third party services, especially for vast datasets can introduce significant costs. Moreover, these services often come with rate limits, potentially slowing down bulk processing tasks.

2.4.3 Natural Language Processing (NLP)

NLP tools, particularly Named Entity Recognition (NER), is widely used for the extraction and categorization of structured information from unstructured text. However, when applied to Pakistani addresses, several challenges arise. Firstly, there is lack of comprehensive training data. High-quality NLP models, especially those using deep learning, require vast amounts of annotated data. The lack of extensive and diverse datasets for Pakistani languages and addresses can hinder the effectiveness of these models. We had tried NER for extraction of components like House #, Building Name, Street, Road, etc. from the address but the NER model was giving near to none accuracy as it could not distinguish between different fields.

Section 3

User Guide

3.1 Code Base

GitHub Repository: https://github.com/haider-06418/Data-Preprocessor

3.2 File Directory

List of main files associated with the project.

1. *normalization.py*: Main file - parsing and normalization of the data.
2. *config.py*: Settings - file paths.
3. *data_preprocessor.py*: Functions for data preprocessing - string manipulation, standardization, tokenization & file handling.
4. *data_processor.py*: Functions for data processing - address fields classification and analysis.
5. *fields_seperation.py*: Functions for fields separation - address fields separation of misplaced fields due to missing comma leading to inaccurate tokenization.
6. *building_extraction.py*: Functions for building name extraction: extracting building names using a corpus
7. *abbreviations.json*: Mappings for common abbreviations.
8. *buildings_directory.txt*: An extensive building name directory.

3.3 Pre-requisites

Pre-requisites for running the program:

1. Install the latest version of Python 3 and PIP
2. Install pandas Python library.
3. Install fuzzywuzzy Python library.

4. Install nltk Python library.

5. Clone GitHub repository to your local machine.

3.4 Running Program

Follow the following steps to use the program:

1. Ensure that the data is a csv file (.csv format).

2. In the file named *config.py*, set the following settings:

   (a) **fname**: Path of the data containing the data enclosed in ‘’. For eg:
   fname = ‘data/tickets/khi_tickets_2022.csv’

   (b) **fname_normalized**: Path of the file (.csv type) containing the normalized data enclosed in ‘’. For eg:
   fname_normalized = ‘data/tickets/normalized_khi_tickets_2022.csv’

   (c) **fname_normalized_excel**: Path of the file (.xlsx type) containing the normalized data enclosed in ‘’. For eg:
   fname_normalized_excel = ‘data/tickets/normalized_khi_tickets_2022.xlsx’

   (d) **columns_to_drop**: Names of the columns which are irrelevant and want to drop. For eg:
   columns_to_drop = [‘Title’, ‘Created’, ‘Close Time’, ‘Queue’]

3. Open CMD/Terminal.

4. Navigate to the folder where the code base is present using the *cd* command.
   For example if the code is present in a folder named Data_Preprocessing in a folder named Work on Desktop, the following command will navigate to the desired folder:
   cd Desktop/Work/Data_Preprocessing

5. Run the program using the following command: *python normalization.py.* While this command is standard, the *python* in the command is subject to your local machine and OS. For python version 3.11 its *py.* For running on a mac its *python3.*

6. After running, processed data will be stored in the location specified in *config.py.*

Section 4

Results

4.1 Normalization Success Rate

The program has a success rates of 95% (± 2 %). This means 95% of the addresses are successfully normalized while a smaller 5% of addresses present anomalies which lead to incorrect field placement. This accuracy was benchmarked against the same dataset but normalized manually containing 75,000+ unique addresses, providing a direct comparison. This indicates the programs's capability to consistently transform diverse address formats into the predefined structure.

4.2 Efficiency

The automation led to a marked reduction in manual intervention and human errors. For perspective, while a manual process for 75,000 unique addresses would take 8-10 hours, our system reduced it to about 10 minutes. This substantial reduction underscores both the speed and the operational advantages of automating the process.

4.3 Scalability

In a scalability test with 215,000+ addresses, the system showcased its ability to handle vast datasets without a hitch. Rigorous testing was undertaken on the dataset of 215,000+ addresses, ensuring a broad spectrum of address formats were considered, thus adding robustness to our evaluation.

4.4 Building Name Extraction

Leveraging the Levenshtein distance methodology, our system achieved an astounding accuracy of 98% in the extraction of building names. The myriad of challenges associated with building names be it misspellings, diverse representations, or varying nomenclature—emphasizes the robustness and adaptability of our approach.

4.5 Address Type Classification

The system's classification capabilities stood out, flawlessly distinguishing between house and apartment types, reaching a 100% accuracy rate.

4.6 Comparison with Alternative Methodologies

Compared to the previously mentioned machine learning solution tailored for Islamabad-based addresses in section 2.4.1 (with an accuracy of 83%), our approach demonstrates superior performance, especially when considering addresses from diverse regions of Pakistan.

4.7 Address Challenges

The nature of manually-written addresses brought forth issues such as spelling errors, diverse abbreviations and inconsistencies. Despite these obstacles, the system displayed adaptability and precision in its address normalization.

Section 5

Conclusion

In a landscape where data is undeniably the driving force behind effective business decision-making, the importance of accurate and standardized address information cannot be overstated. The urgency is amplified in contexts like that of Pakistani addresses, which had long lacked a dedicated normalization system. This project successfully addressed this gap, introducing a tailored and automated system that transforms a myriad of address formats into a consistent and standardized structure. Through the automation of traditionally manual processes, we have not only streamlined data handling but also significantly reduced errors and inconsistencies that were once inevitable. Tackling challenges such as creating a uniform nomenclature and managing the varied nature of addresses, the system now paves the way for enhanced data quality. This breakthrough in address normalization not only simplifies address-based search and analysis but also stands as a testament to the power of automation and innovative thinking in enhancing data quality and, by extension, business intelligence. As we conclude, it is evident that our endeavor has set a new benchmark for address management in Pakistan, and we believe that similar approaches can be replicated in other contexts where data normalization remains a challenge.