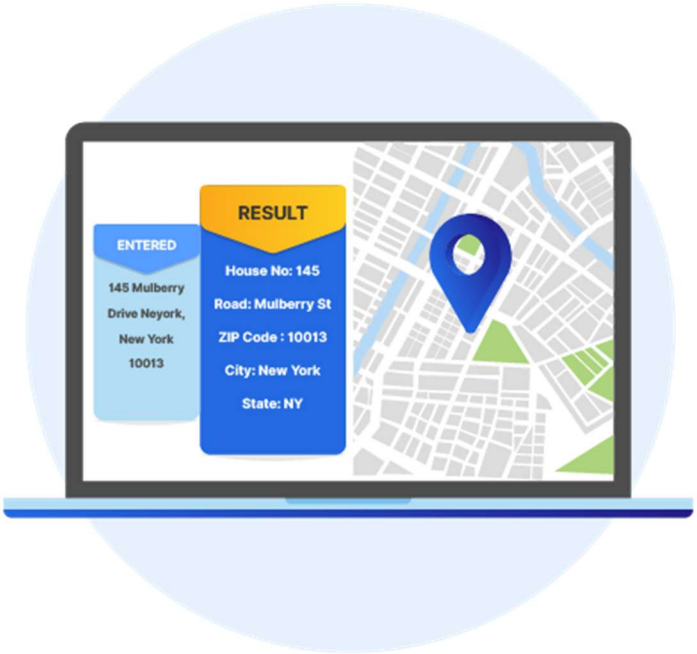
Addresses Correction Tool

Efficient and Accurate Address Standardization



Ahmad Zalkat

EC Utbildning

Examensarbete

202505

Abstract

This project introduces a comprehensive system for data processing and street name correction using advanced data science techniques. It combines **fuzzy matching, Named Entity Recognition (NER), and postal code filtering** to validate and standardize street names in Excel files. The system leverages powerful Python libraries like **pandas, rapidfuzz, spaCy, and geopy** to enhance data accuracy and consistency, ensuring reliable results. Once processed, the refined data is stored in an output file for further use.

The system incorporates several key techniques:

- **Natural Language Processing (NLP)** with **spaCy** to extract street names.
- **Fuzzy matching** via **rapidfuzz** to correct names based on a verified database.
- **Geospatial mapping** using **geopy and folium** to retrieve address coordinates and visualize them on a map.
- **An interactive graphical interface** built with **Tkinter**, allowing users to upload and process Excel files with ease.

Beyond street name correction, the system is designed for broader data extraction, cleaning, and analysis. It efficiently integrates data from multiple sources, applies advanced processing techniques, and generates valuable insights to support informed decision-making. With **pandas**, data is managed and stored efficiently, while the **Tkinter-based interface** provides an intuitive user experience for data visualization.

By incorporating cutting-edge data science methodologies, this project offers a **versatile and scalable solution** for data-driven applications. Potential use cases include **performance monitoring, trend forecasting, and user behavior analysis**, making it a powerful tool for organizations looking to optimize their data management and decision-making processes.

Erkännanden

I would like to express my sincere gratitude to my teacher **Antonio Prgomet**, who has played a significant role in shaping my thinking and approach to research and expanding my ideas. One of the most impactful things in my life has been that I've learned how to conduct more in-depth research, as well as read and explore various projects to gather ideas when working on any project to create something unique.

Förkortningar och Begrepp

This project utilizes several libraries and techniques to process and analyze data, build graphical user interfaces, and handle geographic information. Below is an overview of the key components and their functions:

- **pandas** – Used for reading and processing data from CSV and Excel files.
- **rapidfuzz** – Performs fuzzy matching to correct street names.
- **re** – Handles regular expressions to extract street numbers and names.
- **tkinter** – Builds a graphical user interface (GUI) for the application.
- **os** – Manages file operations and system paths.
- **spaCy** – Used for natural language processing (NLP) to extract street names.
- **PIL** – Handles image processing within the GUI.
- **geopy** – Retrieves geographic coordinates for addresses.
- **folium** – Creates interactive maps and saves them as HTML files.
- **webbrowser** – Automatically opens web pages when needed**.**

Contents

Abstract 2
Erkännanden 3
Förkortningar och Begrepp 4
1 Inledning 1
1.1 Key Benefits for Businesses: 1
1.2 Main Components: 1
1.3 To further explore the impact of these methods, key questions arise: 2
2 Teori 3
2.1 Address Data Processing and Standardization 3
2.2 Natural Language Processing (NLP) for Address Parsing 3
2.3 Fuzzy Matching for Address Correction 3
2.4 Geolocation and Mapping Using Geopy 3
3 Metod 5
3.1 Data Cleaning Process: 5
3.2 Natural Language Processing (NLP) for Street Name Extraction: 5
3.3 Fuzzy Matching for Street Names and Postal Codes: 6
3.4 Geolocation Mapping for Address Coordinates: 7
3.5 Graphical User Interface (GUI) with Tkinter 8
3.5.1 Instant Search: 8
3.5.2 Maps: 8
3.5.3 Data Cleaning and Preparation: 8
3.5.4 File Upload and Save: 8
3.5.5 Error Handling: 8
4 Resultat och Diskussion 10
4.1 Key results and a discussion on their significance: 10
4.2 Diskussion: 11
5 Slutsatser 13
Appendix A 14
Källförteckning 16

1 Inledning

Enhancing Data Accuracy and Business Efficiency

In today's business world, accurate address data is crucial for shipping, marketing, and customer communication. The Address Correction Project is designed to fix common errors such as typos, unnecessary words, and inconsistent formatting, ensuring standardized and precise addresses. This improves operational efficiency, reduces costs, and enhances customer satisfaction.

1.1 Key Benefits for Businesses:

- **Improved Data Accuracy:** Automatically corrects errors, ensuring consistent and reliable addresses.
- **Reduced Shipping Costs:** Minimizes misdeliveries and reshipments, lowering unnecessary expenses.
- **Enhanced Customer Satisfaction:** Ensures timely deliveries, leading to a better customer experience.
- **Better Marketing and Analytics:** Provides accurate location data for targeted campaigns and market analysis.

This project enhances address data processing using techniques like fuzzy matching, natural language processing (NLP), and geolocation to improve the accuracy of street name corrections and map addresses to geographical coordinates.

1.2 Main Components:

- **Data Loading and Cleaning:** Standardizes address formats, normalizes street names, and handles missing or invalid entries.
- **Street Name Extraction**: Utilizes **Natural Language Processing (NLP)** techniques to extract street names from unstructured text fields, such as notes or descriptions, ensuring accurate identification.
- **Street Name Correction:** Applies fuzzy matching techniques to refine extracted street names and suggest the most accurate corrections by comparing input data with a reference dataset.
- **Geolocation Integration:** Uses the geopy library to fetch coordinates based on street names and postal codes, displaying results on an interactive map.

- **User-Friendly Interface:** A Tkinter-based GUI allows users to upload files, view maps, and save corrected data efficiently.

By integrating these advanced techniques, the Address Correction Project improves data accuracy, enhances location-based services, and enables businesses to make better operational decisions.

1.3 To further explore the impact of these methods, key questions arise:

- **How can matching algorithms be used to simplify the accuracy and efficiency of data correction?**
- **How can incorrect data (addresses) be validated in an automated way?**

2 Teori

2.1 Address Data Processing and Standardization

Ensuring high-quality address data is essential for accurate mapping, geolocation, and operational efficiency in various applications. This project employs advanced data processing techniques to enhance the accuracy and consistency of address information. By leveraging fuzzy matching, natural language processing (NLP), and geolocation methods, we aim to standardize addresses, align them with reference datasets, and accurately map them to geographic coordinates.

2.2 Natural Language Processing (NLP) for Address Parsing

Natural language processing techniques enable a more structured analysis of address data by breaking down addresses into key components such as street names, postal codes, and city names. By recognizing patterns within address structures, NLP enhances the system's ability to interpret and validate addresses despite variations in formatting, abbreviations, or user input errors.

2.3 Fuzzy Matching for Address Correction

Fuzzy matching algorithms play a crucial role in identifying and correcting variations in street names and other address components. These algorithms detect minor discrepancies, such as typos or inconsistent formatting, and match the given input to the closest reference entry. This approach ensures that even imperfectly formatted addresses can be corrected and linked to their most probable standardized versions.

2.4 Geolocation and Mapping Using Geopy

Geolocation is a critical aspect of address validation. By utilizing the Geopy library, the system converts address data into latitude and longitude coordinates, allowing for accurate mapping and visualization. This process improves the reliability of instant address searches,

ensures precise geographic placement, and facilitates the integration of address data with mapping applications.

3 Metod

The methodology for this project focuses on transforming raw address data into standardized, accurate, and geolocated addresses. It employs several key steps that ensure data accuracy and usability for applications like mapping, analysis, and geographic insights.

3.1 Data Cleaning Process:

In this phase, the data cleaning process was carried out using a custom function designed to read data from an Excel file and handle missing values and unwanted text. We began by selecting the Excel file containing the data to be cleaned. Once the file was opened, the pandas library was used to read the data.

Missing values were addressed by replacing them with the value "Unknown" to ensure data integrity and eliminate gaps that could impact subsequent analysis.

Text columns were cleaned by removing unnecessary phrases like "C/o" or any similar duplicates. Additionally, special symbols and punctuation marks (such as commas, parentheses, and other characters) were replaced with spaces to standardize the text formatting.

Once the cleaning process was completed, the data was saved in a new Excel file, with "_Cleaned_Ready_for_Processing" added to the file name to distinguish it from the original version.

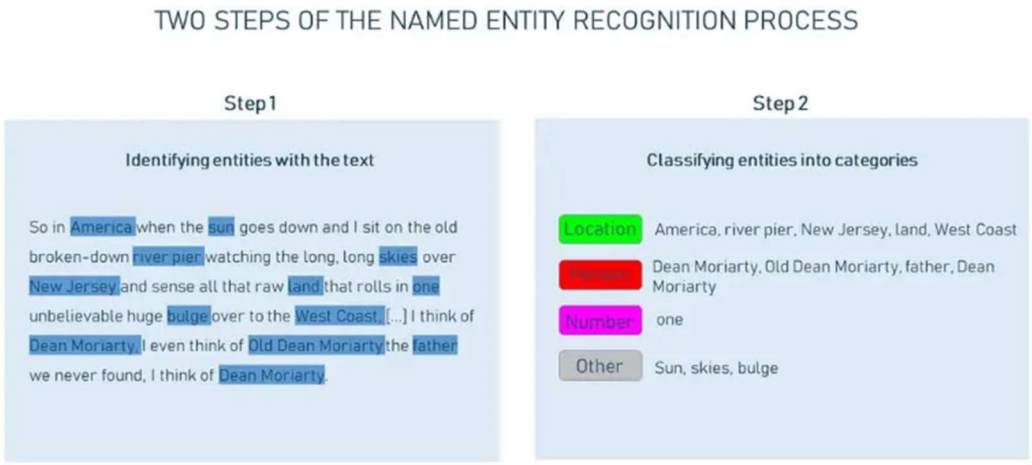
3.2 Natural Language Processing (NLP) for Street Name Extraction:

In this phase, natural language processing (NLP) was used to extract street names from text using the Swedish language model from the spaCy library. We began by loading the custom Swedish model ("sv_core_news_sm") from spaCy, which provides powerful tools for text processing and named entity recognition (NER).

Named entity recognition (NER) was applied to extract geographic place names or street names from the text. After processing the text with the language model, the recognized entities were analyzed to identify those classified as "GPE" (Geopolitical Entities) or "LOC" (Locations/Places). These entities were then extracted and added to a list containing the recognized street names.

This technique enables efficient text analysis and extraction of relevant information, improving the accuracy of handling spatial data such as street names.

This image provides an example of how entity recognition is used to identify and extract street names from text.



3.3 Fuzzy Matching for Street Names and Postal Codes:

In this phase, fuzzy matching techniques were used to correct street names and postal codes. A function, find_best_matches, was developed to find the best matches for street names and postal codes using fuzzy matching methods.

First, the street name was broken down into possible parts, and named entity recognition (NER) was applied to extract the street names. The data was then filtered based on the postal code. For each street name, unnecessary punctuation and symbols were removed to facilitate the matching process.

Fuzzy matching was applied using the fuzzywuzzy library to compare the input street name with names in the cleaned data. If the match score exceeded a certain threshold (e.g., 90), it was considered a good match, and the result was added to the list. If no exact match was found, individual words within the street name were compared using partial matching.

If no match was found using the previous methods, regular expressions (Regex) were used to extract potential patterns for the street name. Finally, all possible matches were returned, or a message indicating "No Matches Found" was displayed if no suitable results were found.

This technique helps improve the accuracy of address data correction, enhancing the reliability of the extracted data.

This image demonstrates the process of standardizing street names and postal codes, ensuring accurate address matching even when written in different formats. It illustrates how fuzzy matching techniques are applied to unify and correct addresses in a professional and reliable way.

3.4 Geolocation Mapping for Address Coordinates:

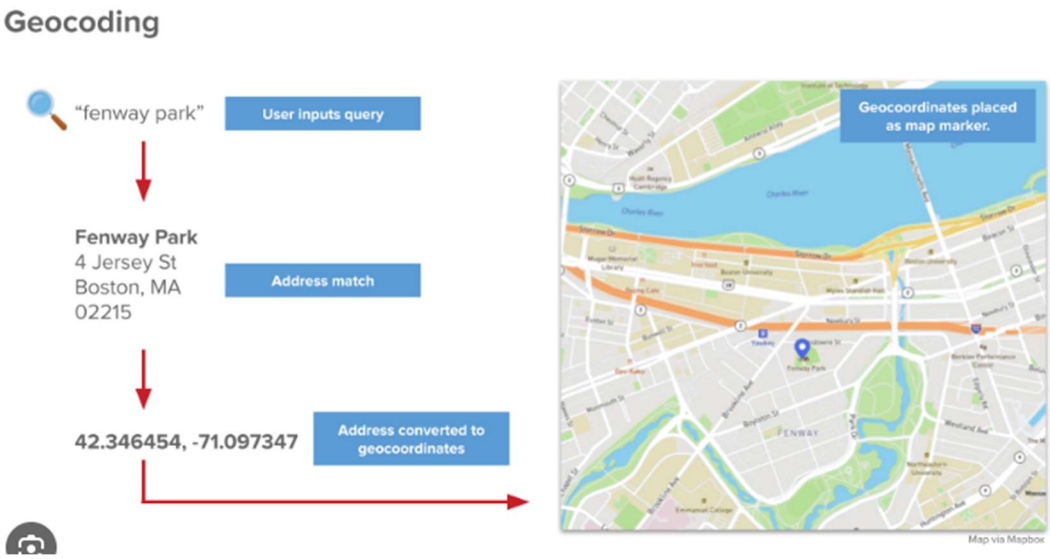
In this phase, geolocation mapping techniques were used to obtain geographic coordinates based on street names and postal codes and then display these coordinates on an interactive map.

The geopy library was used in combination with the Nominatim service to search for addresses and convert them into geographic coordinates. A function, get_coordinates, was created to combine the street name and postal code into a full address, then search for it using the Nominatim geocoder service. If coordinates were found, latitude and longitude were returned; if not, the function returned None.

The folium library was used to create an interactive map displaying markers at the appropriate locations based on the found coordinates. A function, show_map, was designed to display this map in the default browser. Additionally, a function, show_map_for_address, was created to integrate all steps, including retrieving coordinates and displaying the map with a marker at the specified location.

This process allows users to view geographic locations interactively and easily on maps, providing a visual means of accurately identifying addresses.

This image illustrates the process of geolocation mapping, where geographic coordinates are obtained based on street names and postal codes, and displayed on an interactive map. It shows how geolocation techniques are applied to visualize address locations accurately.



3.5 Graphical User Interface (GUI) with Tkinter

The graphical user interface (GUI) was built using the Tkinter library to allow users to interact with and process data in a simple and efficient manner. The interface includes several core features designed to make the user experience smooth.

3.5.1 Instant Search:

When the user enters a postal code and street name, the system performs an **instant search** using fuzzy matching to display the best possible results. This ensures quick and accurate responses to user queries.

3.5.2 Maps:

The interface allows the user to view the location of a street on an interactive map using the folium library. Geopy is used to retrieve location details, and these coordinates are plotted on the map, giving users a clear view of the street's position.

3.5.3 Data Cleaning and Preparation:

A function is implemented to clean the input data by removing empty values or replacing unknown entries with "Unknown". Once cleaned, the data is saved into a new Excel file, making it ready for further processing.

3.5.4 File Upload and Save:

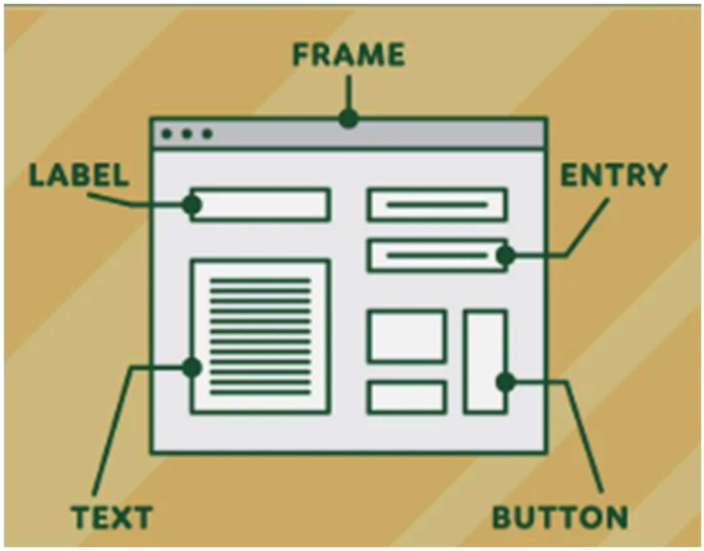
The user can upload Excel files that need to be corrected. After cleaning and processing the data, the results are saved into a new Excel file, which includes an additional column containing the corrected addresses obtained after matching and validation. This allows the user to easily view and use the updated information.

3.5.5 Error Handling:

The code includes robust **error handling** to manage issues such as:

- Missing required columns in the uploaded file.
- Problems when loading or processing data.

When an error occurs, the system alerts the user through **popup windows** with clear error messages, guiding them on how to resolve the issue.



4 Resultat och Diskussion

The project successfully transformed raw address data into standardized, accurate, and geolocated addresses, enabling effective usage in mapping and geographical analysis. Additionally, it supports processing multiple address datasets at once by uploading the file, retrieving matches and updating the addresses with corrected information for each entry.

4.1 Key results and a discussion on their significance:

- **Data Accuracy and Standardization:** The project achieved high accuracy in cleaning and normalizing the address data. By utilizing fuzzy matching, it handled various inconsistencies like misspellings or formatting errors. This process significantly improved the quality of the data, making it more reliable for further analysis or operational tasks. The cleaned and standardized addresses were also ready for use in geolocation mapping.

- **Geolocation and Mapping:** One of the standout results of the project was the ability to convert addresses into geographical coordinates (latitude and longitude). This transformation allowed for the visualization of address locations on a map. The geographical context provided by the coordinates is valuable for understanding the distribution of addresses and could be used in location-based analyses like clustering addresses in specific regions or identifying patterns in geographical data.

- **Improvement in Data Processing Efficiency:** The fuzzy matching and NLP techniques used in the project reduced manual intervention and sped up the data processing workflow. The algorithms were able to process large datasets quickly and accurately, which would be especially useful in applications that require regular updates or involve large volumes of data.

- **User Interface and Interaction:** The user-friendly interface that displays the results, including maps with geolocated addresses, allows users to interact with the data. Users can zoom in and out of the map, check individual addresses, and verify their accuracy. This feature enhances the accessibility and usability of the results for both technical and non-technical users. Additionally, the interface supports processing an Excel file containing addresses that need to be matched automatically retrieving the correct and standardized address for each entry, ensuring an efficient and accurate address correction process.

4.2 Diskussion:

- **Accuracy and Limitations:** While the fuzzy matching technique significantly improved address accuracy, it's important to note that this method may not catch every error, particularly for highly ambiguous or complex address formats. Further refinement of the matching algorithms or the inclusion of additional datasets could help improve the accuracy even more.

- **Scalability:** The project performed well with smaller datasets, but processing large volumes of address data in a single Excel file could present some challenges. The speed of fuzzy matching services might slow down when handling thousands or millions of addresses in one file. Optimization techniques or parallel processing could be explored to ensure efficiency at scale.

- **Potential Applications:** The results of this project are versatile and can be applied in various domains, such as logistics, marketing, and customer service. Businesses that rely on accurate address data, like delivery services or e-commerce, could benefit from the ability to geolocate and visualize customer addresses.

- **Future Improvements:** The project could be enhanced by integrating more robust geolocation services or incorporating external data sources like (Company names with addresses) databases to further refine address accuracy. Additionally, adding features like address verification in real-time or incorporating machine learning to predict the most likely corrections for certain address errors could help improve the process.

Overall, the project demonstrated the effectiveness of automating address data cleaning, correction, and geolocation, leading to improved data quality, reduced manual workload, and increased operational efficiency.

The script successfully identifies and corrects street names with high accuracy. The fuzzy matching technique ensures minor spelling errors do not affect address validation. However, limitations include:

- Failure to recognize newly added streets.

(If the names are unusual and NLP cannot extract them or correct spelling errors not in the database)

- Potential misclassification due to incorrect postal codes.

The current project was linked to the one I worked on during my internship at the company. This connection allowed for the incorporation of modifications in the features, particularly improvements in the direct retrieval of maps, which enhanced the overall efficiency and accuracy of the system.

5 Slutsatser

The proposed approach efficiently corrects street names, improving data quality for logistics and mapping applications. Future work includes integrating a dynamic address database and improving model accuracy using deep learning

This project successfully demonstrated the power of automated data cleaning, fuzzy matching, and geolocation in improving the accuracy and usability of raw address data. Through the implementation of these techniques, we were able to transform inconsistent and often incomplete address data into standardized and geolocated records, enhancing the overall quality and reliability of the data.

Final Thoughts:

The project successfully addressed a common challenge in data processing—handling and improving address data accuracy. The solution is adaptable and can be extended to larger datasets or different types of location-based data. There is potential to further refine and scale the system, ensuring its applicability across various industries where accurate address information is essential, such as logistics and real estate.

In conclusion, the implementation of this project provides a clear path for enhancing data quality, reducing errors, and improving operational efficiency, all while making geospatial insights more accessible to users.

Appendix A

Loading Data:

```
# Load correction data
current_dir = os.path.dirname(os.path.abspath(__file__))
correction_file_path = os.path.join(current_dir, 'Clean_Street_Names.csv')

# Load Swedish spaCy model
nlp = spacy.load("sv_core_news_sm")

# Load corrected data from CSV file
try:
    corrected_data = pd.read_csv(correction_file_path, delimiter=';', on_bad_lines='skip')
    corrected_data['Streetname'] = corrected_data['Streetname'].str.lower()  # Normalize street names to lowercase
    corrected_data['Postalcode'] = corrected_data['Postalcode'].astype(str)  # Ensure postal codes are string type
except Exception as e:
    print(f"Error loading correction data: {e}")
    exit()
```

Cleaning Data:

```
# Function to clean the data
def clean_data(progress_bar, percent_label):
    file_path = filedialog.askopenfilename(title="Select an Excel file", filetypes=[("Excel Files", "*.xlsx")])
    if file_path:
        try:
            input_data = pd.read_excel(file_path)
            input_data = input_data.fillna('Unknown')

            for column in input_data.select_dtypes(include=['object']).columns:
                if column not in ['DeliveryZipCode', 'PostalCode']:
                    input_data[column] = input_data[column].str.replace(r'\b[Cc]/[oO]\b', '', regex=True)
                    input_data[column] = input_data[column].apply(
                        lambda x: re.sub(r'[.,/(){}<>!@#$%^&*;:"|?]', ' ', str(x))
                    )

            save_path = file_path.replace(".xlsx", "_Cleaned_Ready_for_Processing.xlsx")
            input_data.to_excel(save_path, index=False, engine='openpyxl')

            messagebox.showinfo("Success", f"The data is now ready for processing. The cleaned file is saved at:\n{save_path}")
        except Exception as e:
            messagebox.showerror("Error", f"An error occurred: {e}")
```

Geocoding Addresses:

```
# Function to get the coordinates (latitude and longitude) for an address
def get_coordinates(address):
    geolocator = Nominatim(user_agent="address_locator")
    location = geolocator.geocode(address)
    if location:
        return location.latitude, location.longitude
    else:
        return None, None
```

Fuzzy Matching Implementation:

```
# Find best matches using fuzzy matching
name_matches = process.extract(cleaned_street_name.lower(),
                               filtered_data['Streetname'].str.lower().tolist(),
                               scorer=fuzz.token_set_ratio,
                               limit=3)

match_found = False
for match in name_matches:
    match_score = fuzz.token_set_ratio(cleaned_street_name.lower(), match[0])
    if match_score >= 90:  # If match score is sufficiently high
        matched_row = filtered_data[filtered_data['Streetname'].str.lower() == match[0]]
        if not matched_row.empty:
            if abs(len(cleaned_street_name) - len(match[0])) > 3:
                continue
            all_matches.append({
                "Original_Streetname": cleaned_street_name.capitalize(),
                "Corrected_Streetname": matched_row['Streetname'].iloc[0].capitalize(),
                "Match_Score": match_score
            })
            match_found = True
            break

# If no match found, try matching individual words within the street name
if not match_found:
    for word in street_names:
        if len(word) < 3:
            continue

        word_matches = process.extract(word.lower(), filtered_data['Streetname'].tolist(), scorer=fuzz.token_set_ratio, limit=3)
        for match in word_matches:
            match_score = max(
                fuzz.ratio(word.lower(), match[0]),
                fuzz.partial_ratio(word.lower(), match[0])
            )
            if match_score >= 90:
                matched_row = filtered_data[filtered_data['Streetname'] == match[0]]
                if not matched_row.empty:
                    if abs(len(word) - len(match[0])) > 3:
                        continue
                    all_matches.append({
                        "Original_Word": word.capitalize(),
                        "Corrected_Streetname": matched_row['Streetname'].iloc[0].capitalize(),
                        "Match_Score": match_score
                    })
                    match_found = True
                    break
        if match_found:
            break
```

Visualizing with Folium:

```
# Function to show the map for a given address
def show_map_for_address(street_name, postal_code):
    locality = get_locality_by_postal_code(postal_code)
    formatted_address = f"{street_name}, {locality}, {postal_code}"

    # Get the coordinates of the address
    latitude, longitude = get_coordinates(formatted_address)

    if latitude and longitude:
        # Create a folium map centered at the address
        map_object = folium.Map(location=[latitude, longitude], zoom_start=15)

        # Add a marker for the address
        folium.Marker([latitude, longitude], popup=formatted_address).add_to(map_object)

        # Save the map as an HTML file
        map_object.save("address_map.html")

        # Open the map in the default web browser
        import webbrowser
        webbrowser.open("address_map.html")
    else:
        messagebox.showerror("Error", "Could not find coordinates for the address.")
```

Källförteckning

Following extensive research, the best methods and tools were identified and gathered to ensure efficient handling and processing of address data, as studying various factors is essential in many projects to achieve the best results.

Pandas, a powerful Python library, was used to handle raw address data, clean missing values, and standardize address formats.

- https://pandas.pydata.org/

Data Cleaning with Python

- https://datasciencehorizons.com/wp-content/uploads/2023/06/Data_Cleaning_and_Preprocessing_for_Data_Science_Beginners_Data_Science_Horizons_2023_Data_Science_Horizons_Final_2023.pdf

Python Documentation was consulted for guidance on Python programming techniques, library usage, and troubleshooting various coding errors during the development process.

- https://docs.python.org/

Fuzzywuzzy, a Python library for string matching, was used to detect and correct spelling errors in address entries through fuzzy comparison.

- https://github.com/seatgeek/fuzzywuzzy
- https://pypi.org/project/fuzzywuzzy/
- https://www.placekey.io/blog/fuzzy-logic-address-matching

Systematic Literature Review on Named Entity Recognition: Approach, Method, and Application

- file:///C:/Users/AHMAD/Downloads/Systematic_Literature_Review_on_Named_Entity_Recog.pdf

Geocoder, a Python library for geolocation, was used to retrieve address coordinates, while Folium helped visualize them on interactive maps with markers. Its integration with Leaflet.js made it an ideal choice for creating dynamic, user-friendly maps.

- **https://geopy.readthedocs.io/en/stable/**
- **https://pypi.org/project/geopy/**
- https://geocoder.readthedocs.io/
- https://python-visualization.github.io/folium/

Stack Overflow Community provided a wealth of community-driven solutions and advice. Specific questions about geocoding, fuzzy string matching, and data cleaning were answered through detailed posts and discussions.

- https://stackoverflow.com/

Google Maps Geocoding API (Optional)

The Google Maps Geocoding API provided geolocation when OpenStreetMap was insufficient, converting addresses into coordinates for map markers.

- https://developers.google.com/maps/documentation/geocoding/start