A Novel Approach to Address Validation using String Distance Metrics

Sahil Kale[1]

Computer Engineering
Pune Institute of Computer Technology
Pune, Maharashtra, India

Abstract - *Address validation holds fundamental value in confirming the accuracy and geographical precision of addresses used by location-dependent and delivery-based organizations. Addresses often suffer from problems such as missing components and geographical inadequacies which can cause grave logistical issues if not validated adequately beforehand. The identification of missing or invalid address components to perform address validation can prove to be a helpful factor in saving time and cost for businesses and simultaneously reducing the chances of errors in service. Significant potential has been found in the usage of statistical measures such as correlation coefficients and measures of central tendency to perform the task of address validation. The system proposed in this paper uses a combination of different string-matching metrics to generate a normalized score based on statistical similarities. This score can then be used to filter out validated addresses according to the threshold of similarity required. Experiments have been conducted on a real-world healthcare dataset to demonstrate the effectiveness of the proposed approach in terms of accuracy and precision.*

Key Words: **Address Matching, Address Validation, Geocoding, Natural Language Processing, String Matching**

1.INTRODUCTION

Addresses are fundamental in pinpointing a geographical location on earth. By improving the accuracy of addresses, considerable savings can be achieved in terms of time and money for organizations that rely on the precision of auto-generated addresses to maximize customer satisfaction [1]. Automatic address generation is often done using the reverse geo-coding process, which converts geographical co-ordinates into textual addresses [2]. Even though the validation of these addresses, i.e., matching them with correct and verified addresses, seems to be a straightforward task, a lot of complications exist while trying to perform the same.

Address validation, the process of verifying the accuracy and precision of an auto-generated address by matching it with a true counterpart, suffers from several problems when faced with unstructured addresses which may have missing attributes or geographic inconsistencies [3]. The problem caused by geographical inconsistencies can be seen in fig.1, which represents how missing elements in the generated address can be difficult to verify by direct comparison.

The main contribution made by this paper includes proposing a system that resolves the issue of address validation by developing a novel algorithm that requires no pre-processing of the input addresses. It uses statistical correlation measures as weights to combine different string-matching metrics and generate a normalized matching score. This matching score is then utilized to filter out the validated addresses and store them for further use.

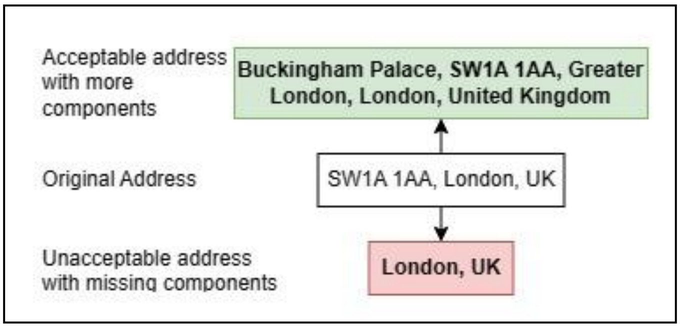
Acceptable address with more components
Buckingham Palace, SW1A 1AA, Greater London, London, United Kingdom
Original Address
SW1A 1AA, London, UK
Unacceptable address with missing components
London, UK

Fig -1: Problems in Address Validation

2.LITERATURE REVIEW

Address validation has historically been dealt with as a pure NLP (Natural Language Processing) task involving sequence labeling. Hidden Markov Models (HMMs) [4] and Conditional Random Field models [5] have been used as a deep learning approach to address validation, but suffer in performance when given inputs of non-standardized addresses. In [6], the authors proposed a method using the BERT language model which can help in contextual modeling of text data. However, none of the existing systems are able to deal with address validation without some preprocessing before moving to the task of validation, which may cause a compromise in precision, a problem that the proposed system aims to solve.

In order to build an efficient architecture for address validation, the system proposed in this paper takes the support of commonly-used string matching metrics to perform the task of address validation. String-matching metrics can be broadly classified into three major types, (i) Edit distance-based metrics, (ii) Token-based metrics and (iii) Hybrid metrics, a detailed explanation of which can be found in [7]. On the basis of the comparisons carried out by [7] and [8], a set of six best performing metrics was chosen to design the methodology followed in this paper.

Levenshtein distance metric, which assigns unit cost to all edit operations [9], and Jaro Winkler distance, which also takes into account transpositions [10], are the edit-based metrics used, while Jaccard distance and Cosine distance are the token-based metrics utilized described in [11]. The Monge Elkan hybrid metric [12] and the Burrows–Wheeler transform distance (BWT) [13] based on string compression have also been used as matching metrics to design the proposed system.

3.METHODOLOGY

3.1 Terminology

1. *Manual Address*: Address fetched from a database and verified to be accurate and precise
2. *Auto-generated Address*: Address generated using reverse geo-coding that is to be validated
3. *Auto Score (Manual Score)*: Normalized score in the range 0 to 1 indicating similarity, where 0 represents complete mismatch and 1 represents a perfect match
4. *Threshold*: Matching score value to be used for filtering out valid auto-generated addresses (default is 0.5)

3.2 System Architecture

As represented in fig. 2, the architecture of the proposed system consists of a database of manual addresses, the corresponding auto-generated addresses, the matching component used to generate a matching score and the filtering component which uses a threshold value to filter out the validated addresses. The addresses to be compared pass through the matching component, which uses a combination of novel algorithms to generate a similarity score based on various string-matching metrics. The process of validation of addresses is then completed by making use of the final component to filter out valid addresses and optionally store them in a database for further processing.

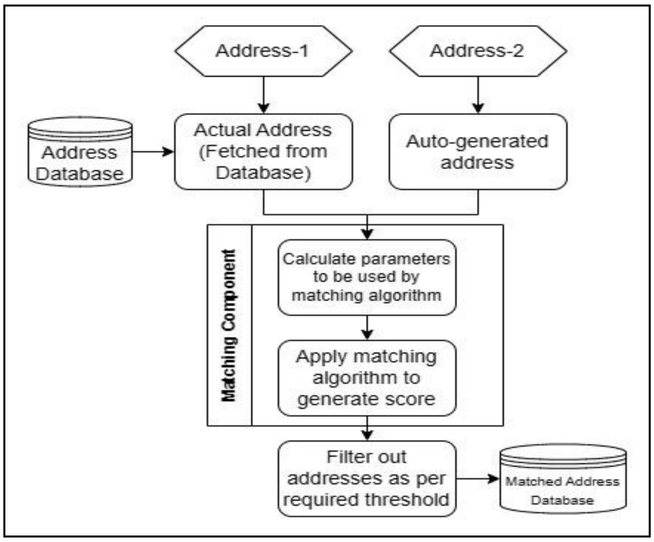
Address-1
Address-2
Address Database
Actual Address (Fetched from Database)
Auto-generated address
Matching Component
Calculate parameters to be used by matching algorithm
Apply matching algorithm to generate score
Filter out addresses as per required threshold
Matched Address Database

Fig -2: System Architecture

3.2 Proposed Methodology

This section provides a brief explanation of the system proposed in this paper, along with a description of the working of its key modules. As represented in the system architecture, the core constituent of the proposed system is the matching component. It generates a normalized matching score by making use of a novel approach using mathematical modeling to combine various string-matching metrics.

The system uses a combination of statistical measures such as Karl Pearson's correlation coefficient and arithmetic mean, mathematical details of which can be found in [14], to achieve the validation task. Algorithm-1 calculates a set of parameters used for generating a matching score in Algorithm-2, which passes the scores on to Algorithm-3 that simply filters out the addresses based on the corresponding threshold.

```
Algorithm-1: Generation of Weights for Matching Metrics
Input:
1. manual_addresses := set of addresses fetched from a database for the given locations, verified to be accurate
2. auto_addresses := set of addresses to be validated which may have been generated using reverse geo-coding
3. metrics := set of string matching metrics to be used, by default 6 metrics are used: [Levenshtein Distance, Jaro Winkler Distance, Jaccard Distance, Cosine Distance, Monge Elkan Distance and BWTRLENCD Distance]

Output:
1. weights : = set of weights to be used for generating matching score
2. scalers := set of numeric values to bring all metrics within the same range

Algorithm:
1. Take a subset of approximately 10 percent of addresses to be validated
2. For each pair of manual address and auto-generated address, assign a manual matching score by following the given process:
    2.1 Start with a baseline score of 0.5
    2.2 For each matching token between manual address and auto-generated address, add a value of 0.1 to the manual score, ensuring that score remains below 1
    2.3 For each mismatched token in the two addresses:
        2.3.1 If the token in manual address adds more specificity and is not present in auto-generated address, subtract a value of 0.1 to the manual score, ensuring that score remains below 1
        2.3.2 If the token in auto-generated address adds more specificity and is not present in manual address, add a value of 0.1 to the manual score, ensuring that score remains below 1
```

3. Using each string-matching metric in set of metrics, calculate normalized distance between each manual address and auto-generated address

4. Calculate the correlation coefficient (corr) between scores generated by each metric and the manual score and store these coefficients as weights

$$weight[metric_i] = corr\ (metric_i\ scores, manual\ score)$$

5. Calculate the difference between mean of manual score and mean of scores generated by each metric and store these differences as scalers

$$scaler[metric_i] = mean\ (manual\ score) - mean(metric_i\ score)$$

6. Pass weights and scalers to the next stage

Algorithm-2: Generation of Normalized Matching Score

Input:

1. manual_addresses and auto_addresses
2. metrics := set of string metrics used in algorithm-1
3. weights : = set of weights generated by algorithm-1
4. scalers := set of numeric values to bring all metrics in the same range generated by algorithm-1

Output:

1. auto_scores := set of matching scores for each pair of addresses in the range 0 to 1, indicating similarity where 0 represents complete mismatch and 1 represents perfect match

Algorithm:

1. Repeat the following steps to generate auto-score for each pair of manual and auto-addresses:
    1.1 Calculate values of normalized distance between each manual address and auto-generated address using given metrics
    1.2 Calculate auto_score by using the formula and add to set of auto_scores.

$$auto\ score = \frac{\sum_{i=0}^{n}(metric\ score + scaler[metric_i]) * weight[metric_i]}{\sum_{i=0}^{n} weight[metric_i]}$$

where, n= number of metrics used
scaler[metric$_i$] = scaler value for metric '*i*' from scalers array
weight[metric$_i$] = weight value for metric '*i*' from weights array

2. Pass the set of addresses and auto_scores to the next component

Algorithm-3: Filtering out validated addresses

Input:

1. auto_addresses := set generated by algorithm-2
2. auto_scores := set generated by algorithm-2
3. threshold := value used for filtering out valid auto-generated addresses (default is 0.5)

Output:

1. set of validated addresses optionally stored in a database

Algorithm:

1. For each address in auto_addresses, repeat the following:
    1.1 If auto_score for that address is greater than threshold, add the address to set of validated addresses
2. Store set of validated addresses in a database for further processing

4. EXPERIMENTATION AND RESULT DISCUSSION

In order to demonstrate and prove the efficiency and accuracy of the system, comprehensive experimentation was carried out using Google Colab. The publicly available 'PMJAY Healthcare Database', details of which can be found on the official website [15], was used as a reference to get the manual addresses for experimentation. It contains a list of addresses of various healthcare centers in India, verified by the government of India.

Reverse geocoding was performed on the co-ordinates of the hospital locations using the reverse_geocoder library in Python to generate a set of auto-addresses to be validated. A sample subset of 98 samples was then taken out of this data and manual scores were assigned to the corresponding auto-addresses. After calculating normalized distances using the six metrics as suggested in the algorithm, the correlation heatmap was obtained as shown in fig. 3.

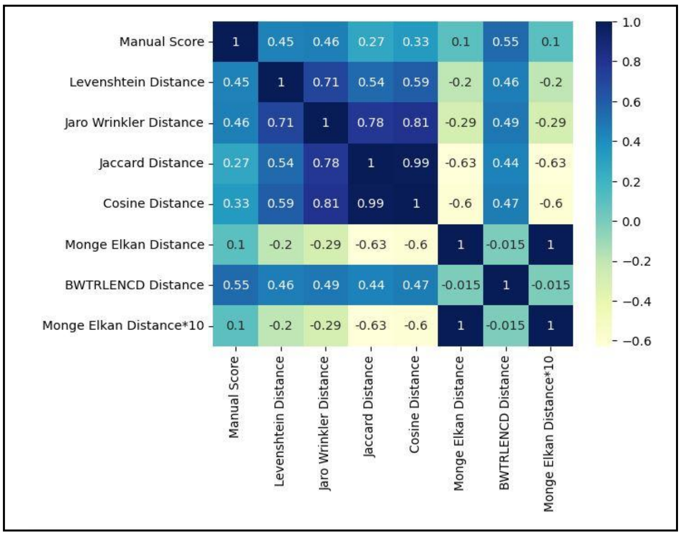

Fig -3: Correlation heatmap between manual score and string-matching metrics

Similarly, the distribution trend of the manual score was compared with the distribution trends for each metric by using line plots as shown in fig. 4. In order to bring the distribution of the scores in the same range, the metric scores are adjusted with scalers, which represent the difference between mean values of the scores and the mean value of the manual score, which was equal to 0.567347. The weights and scalers to be used for the generation of the auto-score were thus obtained in table 1.

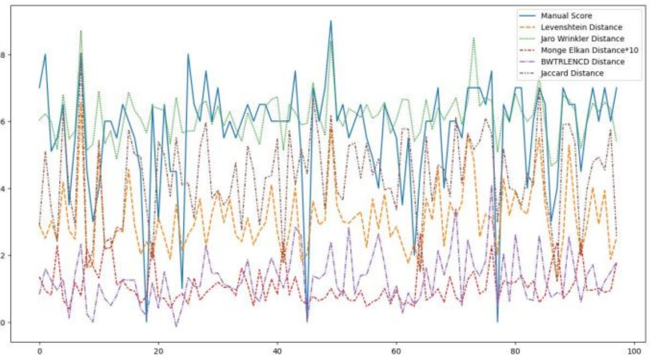

Fig -4: Comparison of the distribution of manual score and scores produced by string-matching metrics

Table -1: Weights and Scalers obtained for each metric

Metric	Weight (Correlation Coefficient)	Scaler (Difference between means)
Levenstien score	0.45	+0.260638
Jaro Winkler Distance	0.46	-0.04845
Jaccard Distance	0.27	+0.135038
Cosine Distance	0.33	-0.058378
Monge-Elkan Distance (x10)	0.1	+0.460816
BWTRLENCD Distance	0.55	+0.438219

The auto-scores were then calculated using the specified formula and the confusion matrix was plotted as shown in table 2, to estimate the accuracy of address validation done by the system.

Table -2: Confusion matrix for obtained results

Actual → Predicted ↓	Accept (manual score >= 0.5)	Reject (manual score < 0.5)	Total
Accept (auto score >= 0.5)	75	9	84
Reject (auto score < 0.5)	1	13	14
Total	76	22	98

The confusion matrix shows that 76 addresses would be filtered out as validated addresses, with only one address wrongly validated as correct. The standard evaluation metrics were then calculated using the confusion matrix to obtain the results displayed in table 3, with a few sample outputs represented in fig. 5. Significant values of precision, which measures the proportion of positive identifications that are actually correct, and recall, that represents the proportion of actual positives correctly identified, were obtained, thus proving the efficiency of the proposed system for the task of address validation.

Table -3: Results of experimentation

Evaluation Parameter	Value (percentage, except F1 score)
Accuracy	89.79591836734694 %
Precision	89.28571428571429 %
Recall	98.68421052631579 %
F1 Score	0.9375

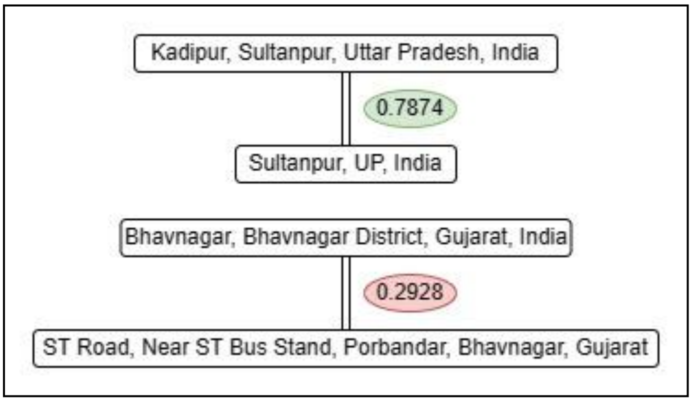
Kadipur, Sultanpur, Uttar Pradesh, India
0.7874
Sultanpur, UP, India
Bhavnagar, Bhavnagar District, Gujarat, India
0.2928
ST Road, Near ST Bus Stand, Porbandar, Bhavnagar, Gujarat

Fig -5: Example scores generated by the proposed system (auto-generated addresses above compared with manual addresses below)

5. CONCLUSION

Address validation, as an NLP task, has often proven to be a long and tedious process that does not have a fixed solution due to the wide variety of techniques available to generate addresses automatically. However, with the help of the system proposed in this paper, address validation can be performed in a simple and efficient manner and can solve problems such as geographical inconsistencies and unformatted inputs to generate accurate matching scores.

A combination of novel algorithms to generate a similarity score based on various string-matching metrics combined using weights as their correlation coefficients and adjusted by mean values has thus been explained and proven to be applicable to the address validation task successfully. Experimental evaluations, carried out on a real-world dataset involving addresses of healthcare centers prove the effectiveness and practicability of these scores generated using the technique provided in this paper.

In the future, this work can be extended by generating an API to make the proposed technique more usable and exploring artificial intelligence-based approaches to simplify the task of manual scoring for the system.

REFERENCES

[1] Coetzee, S.; Cooper, A.K. "Value of addresses to the economy, society and governance—A South African perspective". In: Proceedings of the 45th Annual Conference of the Urban and Regional Information Systems Association (URISA), Washington, DC, USA, 20–23 August 2007.

[2] Geocoding API. Available online: https://developers.google.com/maps/documentation/geocoding/overview (accessed May 2023).

[3] Guermazi, Y., Sellami, S., Boucelma, O. "A RoBERTa Based Approach for Address Validation". In: New Trends in Database and Information Systems. In: ADBIS 2022, Communications in Computer and Information Science, vol 1652. Springer, Cham, 2022.

[4] P. Christen and D. Belacic. "Automated probabilistic address standardisation and verification". In: Australasian Data Mining Conference (AusDM'05), pages 53–67, Sydney, 2005.

[5] M. Wang, V. Haberland, A. Yeo, A. Martin, J. Howroyd, and J. M. Bishop. "A probabilistic address parser using conditional random fields and stochastic regular grammar". In: 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), pp. 225–232, 2016.

[6] H. Zhang, F. Ren, H. Li, R. Yang, S. Zhang, and Q. Du. "Recognition method of new address elements in chinese address matching based on deep learning". In: ISPRS International Journal of Geo-Information, vol. 9, no. 12, p. 745, 2020.

[7] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. "A comparison of string metrics for matching names and records." In: KDD Workshop on Data Cleaning and Object Consolidation, Vol. 3. 73–78, 2003.

[8] Rasool, Akhtar, Amrita Tiwari, Gunjan Singla and Nilay Khare. "String Matching Methodologies: A Comparative Analysis.". In: International Journal of Computer Science and Information Technologies, vol.3 issue 2, pp. 3394 – 3397, 2012.

[9] S. Zhang, Y. Hu and G. Bian, "Research on string similarity algorithm based on Levenshtein Distance". In: 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, pp. 2247-2251, 2017.

[10] Wang, Y., Qin, J., Wang, W. "Efficient Approximate Entity Matching Using Jaro-Winkler Distance". In: Web Information Systems Engineering – WISE 2017. WISE 2017. Lecture Notes in Computer Science, vol 10569. Springer, Cham, 2017.

[11] J. Wang, G. Li and J. Fe, "Fast-join: An efficient method for fuzzy token matching based string similarity join". In: 2011 IEEE 27th International Conference on Data Engineering, Hannover, Germany, pp. 458-469, 2011.

[12] Jimenez, S., Becerra, C., Gelbukh, A., Gonzalez, F. "Generalized Mongue-Elkan Method for Approximate Text String Comparison". In: Gelbukh, A. (eds) Computational Linguistics and Intelligent Text Processing. CICLing 2009. Lecture Notes in Computer Science, vol 5449. Springer, Berlin, Heidelberg, 2009.

[13] Manzini, G. (1999). "The Burrows-Wheeler Transform: Theory and Practice". In: Kutyłowski, M., Pacholski, L., Wierzbicki, T. (eds) Mathematical Foundations of Computer Science. MFCS 1999. Lecture Notes in Computer Science, vol 1672. Springer, Berlin, Heidelberg, 1999.

[14] Springer: Texts in Statistics Book Series. Available online: https://www.springer.com/series/417 (accessed May 2023).

[15] Pradhan Mantri Jan Arogya Yojana (PM-JAY)- Locations and Addresses of Empanelled Hospitals. Available online: https://hospitals.pmjay.gov.in/mapsPlotting.htm (accessed May 2023).