Deep neural network based learning to rank for address standardization

Hai-Nam Cao
Hanoi University of Science and Technology
Email: nam.ch173275@sis.hust.edu.vn

Viet-Trung Tran*
Hanoi University of Science and Technology
Email: trungtv@soict.hust.edu.vn
*Corresponding author

***Abstract*—Address standardization is the process of converting and mapping free-form addresses into a standard structured format. For many business cases, the addresses are entered into the information systems by end-users. They are often noisy, uncompleted, and in different formatted styles. In this paper, we propose a deep learning-based approach to address standardization challenges. Our key idea is to leverage Siamese neural network model to embed raw inputs and standardized addresses into a single latent multi-dimensional space. Thus, the corresponding output is the one with the highest-ranking score. Our experiments demonstrate that our best model achieved 95.41% accuracy, which is 6.6% improvement from the latest SOTA.**

***Index Terms*—address standardization, siamese neural network, learning to rank, sentence embedding, BERT.**

I. Introduction

Address standardization is the process of converting and mapping free-form addresses into a standard structured format. This process's automation is crucial in many business cases, such as in e-commerce, logistics, and geographical data analysis.

For many business cases, the addresses are entered into the information systems by end-users. They are often noisy, uncompleted, and in different formatted styles. For instance, the address string 'đường thái hà thành phố hà nội' does not contain any information about the district. But, the address standardization must give out the complete output. In our example, the output should be {street: thái hà, district: đống đa, city: hà nội}.

Given raw address string, a common approach for address standardization may perform these following steps: (1) recognizing the mentioned entities based on Named Entity Recognition (NER) models in natural language processing, (2) normalizing the recognized entities by mapping to the valid ones, (3) fulfilling the missing fields (e.g., ward, district) according to the predefined standard format. Step (2) and (3) implementations are mostly rule-based. They require a complete set of valid addresses, usually stored in a multiple-column table, namely referenced address table (RAT) as in Table I.

street	ward	district	city
tây sơn	null	đống đa	hà nội
cát linh	null	đống đa	hà nội
null	văn miếu	đống đa	hà nội
...	...	...	...
null	vĩnh an	vĩnh lộc	thanh hóa
null	vĩnh hòa	vĩnh lộc	thanh hóa
null	vĩnh hùng	vĩnh lộc	thanh hóa
...	...	...	...

Table I: Referenced Address Table consists of 4 columns: street, ward, district, city

In this paper, we propose a ranking approach based on deep neural network to address standardization. For a given input address string x, we perform one single step to rank x over every normalized address in the RAT table, and return the one associated with the highest score. Our approach achieved 6.6% improvement over the latest SOTA approach proposed in [1].

The rest of this paper is organised as follows. We review related literature on Section II. Section III details the proposed neural network architecture and various feature engineering optimizations. Section IV discusses the experimental setup and results. We conclude and discuss future direction in Section V.

II. Related Work

Fuzzy matching approach [2], [3], [4] leverage fuzzy search on the RAT table for the best match. These approaches do not require a training phase, but they do not guarantee high accuracy. Chatterjee et al. used *Apache Solr*, an open-source platform, to perform the fuzzy-query and leveraged a candidate graph to score the search result.

NER approaches consider address standardization as information extraction and involve identifying expressions that refer to entities such as street, ward, and city... NER-based models combine NER algorithms and some rules generated to correct the name of detected entities and fulfill missing fields in the NER output. [5], [6], [7], [8] experimented Hidden Markov Models (HMM) to extract address fields. In [9], Wang et al. combine a Conditional Random Field (CRF) model and a post-processing step based on learned stochastic regular grammar (SRG) to enhance the model accuracy. Recently, [10] proposed a multi-layer feed-forward neural network to recognized address fields.

Our previous work proposed Stavia model [1]. Stavia model consists of three blocks: (1) a Fuzzy-matching block is responsible for finding top K potential candidates in RAT; (2) a NER block aims to detect entities and their labels from raw address; and (3) a Re-ranking block a log-linear estimation model to

Raw Address	Standard Address
150 Kim Hoa Hà Nội	{street: kim hoa, district: đống đa, city: hà nội}
phường Lĩnh Nam quân Hoàng Mai, tp Hà Nội	{ward: lĩnh nam, district: hoàng mai, city: hà nội}
tay son dong da	{street: tây sơn, district: đống đa, city: hà nội}

Table II: Examples of Address Standardization

combine outputs of the two previous block to return the highest rank, which corresponds to the standardized address of the input.

III. Deep neural network for address ranking

A. Siamese neural network

Siamese Neural Network (SNN) was first introduced in 1993 by Bromley and LeCun to solve the Signature Verification problem [11]. SNN has been applied to many tasks such as image recognition, image verification, homo-glyph attacks detection, one-shot, and few-shot learning [12], [13], [14], ect.

SNN is suitable to measure the similarity between two inputs. SNN contains two identical parallel sub-networks, both sharing the same weights. Each sub-network accepts a dedicated input. Their weights are updated simultaneously during training. The outputs of two identical networks are combined to make the final prediction.

The training set composes of triplets (x_1, x_2, y), where x_1 and x_2 represent two distinct inputs, and y $\in \{0, 1\}$ indicates that if x_1 and x_2 are similar (y = 1) or dissimilar (y = 0).

B. Siamese neural network for address ranking

Inspired by the particular architecture of SNN and its possibility to map both the two inputs into the same latent embedding space, we design our SNN-based network that can accept the input as a pair of addresses and predicts their similarity. We design our network in three blocks, as follows.

Feature engineering block This block takes the input pair X_1, X_2, which are the raw address and one standardized address in the RAT table. This block aims at initializing the feature vectors. We denote these vectors v_{raw} and v_{std} respectively. Details of the method to encode them will be presented in III-C.

Embedding block This block composes of two identical networks. Each sub-network of this block consists of a sequence of dense layers, sharing the same architecture and weights. These networks accept the v_{raw} and v_{std} as their inputs and output the embedding vectors in the same low-dimensional latent space. We denote these vectors $v_{raw-embedding}$ and $v_{std-embedding}$.

Ranking block This block calculates the similarity score between $v_{raw-embedding}$ and $v_{std-embedding}$. The architecture of the Ranking block consists of a sequence of dense layers. The first layer of this block is the combination of $v_{raw-embedding}$ and $v_{std-embedding}$ to create vector z_1 accounting for interactions between raw address and standard address features. We introduce 4 different combinations, each combination corresponds to a specific model.

- The first model is a Merge model. z_1 is generated by concatenating $v_{raw-embedding}$ and $v_{std-embedding}$.

$$\begin{aligned} z_1 &= \phi(v_{raw-embedding}, v_{std-embedding}) \\ &= \left[\ v_{raw-embedding}, v_{std-embedding}\ \right] \end{aligned}$$

- The second model is an Element-Wise model. We directly use the dot product of $v_{raw-embedding}$ and $v_{std-embedding}$ to create z_1.

$$\begin{aligned} z_1 &= \phi(v_{raw-embedding}, v_{std-embedding}) \\ &= v_{raw-embedding} \odot v_{std-embedding} \end{aligned}$$

- The third model is an Absolute-Difference model. z_1 is defined as:

$$\begin{aligned} z_1 &= \phi(v_{raw-embedding}, v_{std-embedding}) \\ &= |v_{raw-embedding} - v_{std-embedding}| \end{aligned}$$

- The fourth model is the Addition model. z_1 is calculated by algebraic function defined as:

$$\begin{aligned} z_1 &= \phi(v_{raw-embedding}, v_{std-embedding}) \\ &= w_0 \odot v_{raw-embedding} + w_1 \odot v_{std-embedding} \end{aligned}$$

where $\odot$ denotes the element-wise product of vectors, w_0 and w_1 being trainable parameters are randomly initialized with a Gaussian distribution (with a mean of 0 and standard deviation of 0.05). This function is more flexible as compare to Absolute Difference Function.

Since z_1 accounts for interactions between raw address and standard address features. To make this interactions stronger in order for the model to learn the similarity or dissimilarity between two feature vectors better, we propose to add sequences of dense layers on top of z_1 vector. Generally, the architecture of block three is defined as:

$$\begin{aligned} z_1 &= \phi(v_{raw-embedding}, v_{std-embedding}) \\ z_2 &= a_2(W_2^T z_1 + b_2) \\ &\ldots\ldots \\ z_L &= a_L(W_L^T z_{L-1} + b_L) \\ y_{sc} &= \sigma(h^T z_L) \end{aligned}$$

where W_x, b_x, a_x denote weight matrix, bias vector, and activation function for the x-th layer's perceptron, respectively. σ is sigmoid function. The output of SNN is y_{sc}, the similarity score of two input addresses, in range (0,1).

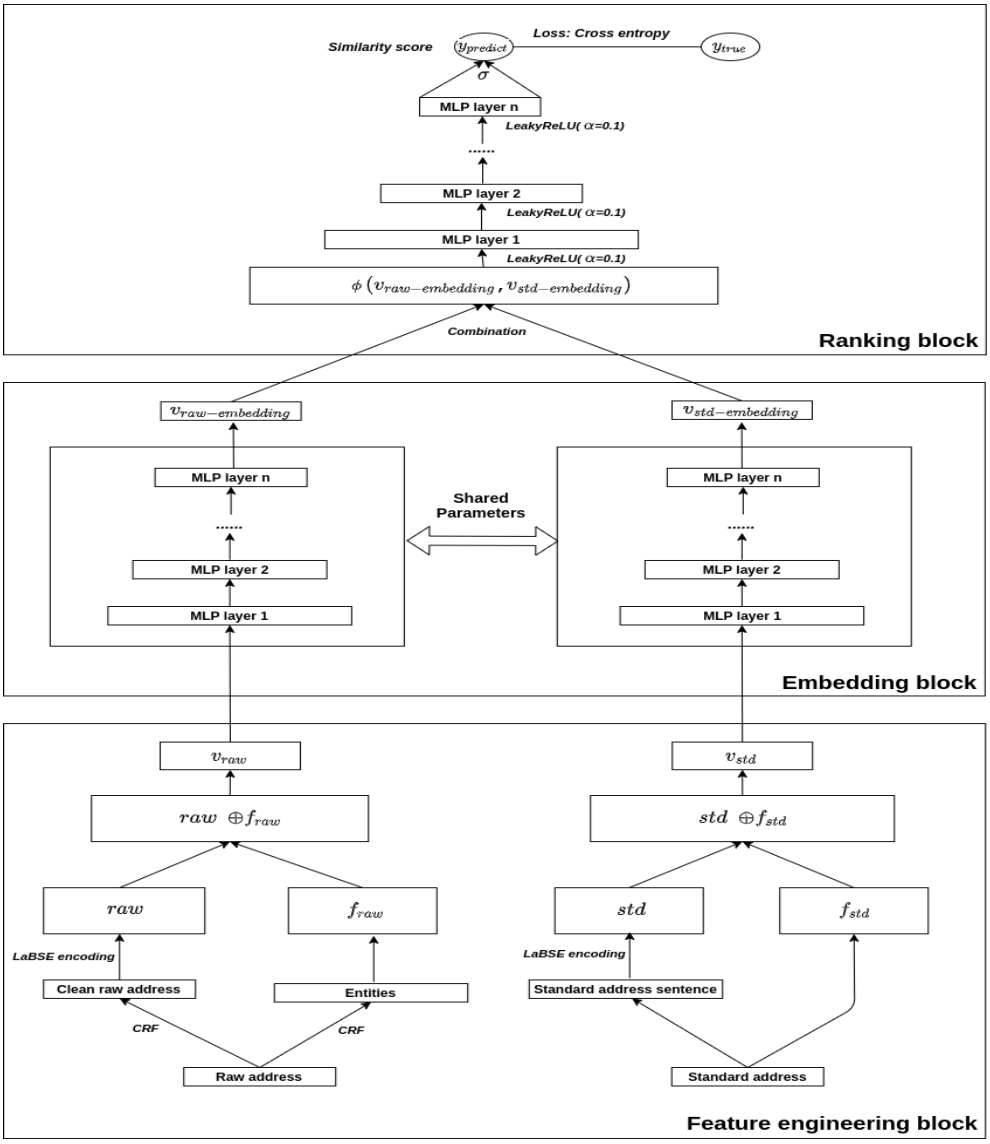

Figure 1: Our proposed neural network architecture

1) Training Phase: To train our model, we feed the network with $[X_1, X_2, Y]$ in each iteration, where X_1 is one raw address, X_2 is a standardized address which belongs to RAT, and Y is the label denoting if X_2 is normalization form of X_1 or not.

As our objective is to classify the similarity between the input addresses, we make use of binary cross-entropy as the model loss function. Binary cross-entropy loss, also called log loss, estimates a classifier's performance with an output probability ranging from 0 to 1. The loss value will grow up in case the predicted probability is far from the true label. The loss can be formulated as follow:

$$L = -ylog\hat{y} - (1-y)log(1-\hat{y})$$

where y and $\hat{y}$ represent the true label and prediction probability, respectively.

2) Inference Phase: In the inference phase, we compute the similarity of the raw address with every standardized address in the RAT table by feed-forwarding the SNN model. The corresponding standardized address for the raw address is the one with the highest score (and bigger than a configured threshold).

Note that the computation does need to be sequential and potentially slow. We take advantage of batch processing in the deep learning framework (e.g., Pytorch and Tensorflow) to rank the raw address over the RAT table in a single forwarding step.

C. Implementation

In this section, we discuss various feature engineering tactics to optimize our model.

1) Enriching RAT with the equivalent non-accent addresses: Particularly, Vietnamese addresses are commonly

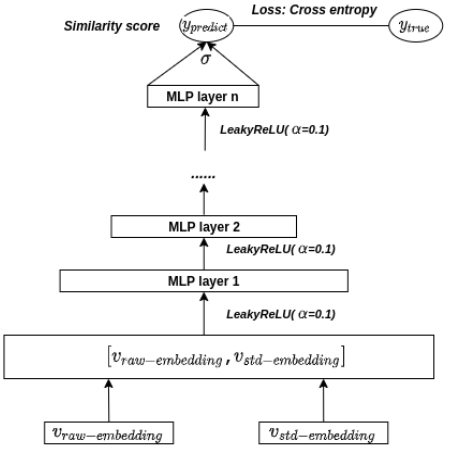

Figure 2: Ranking block architecture: Merge model

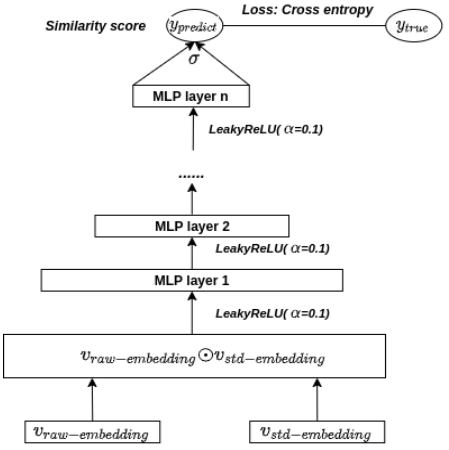

Figure 3: Ranking block architecture: Element-Wise model

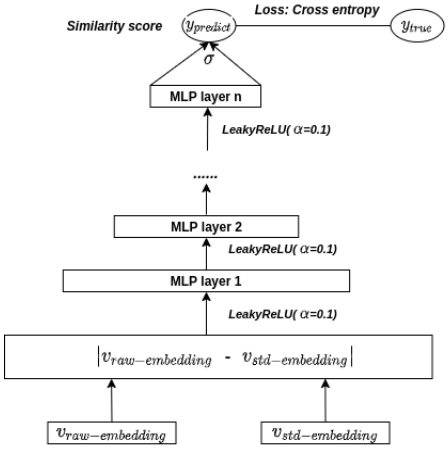

Figure 4: Ranking block architecture: Absolute-Difference model

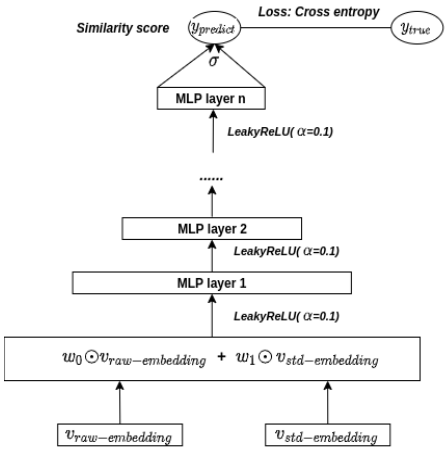

Figure 5: Ranking block architecture: Addition model

written with and without accents. Given the raw input address, which can be in accented or unaccented form, we enrich our RAT table with the equivalent non-accent addresses. If a given raw address is accented, it will be ranked with accented addresses. Vice versa, unaccented addresses will be selected.

2) BERT for address embedding: To encode address string to feature vector, we can leverage existing sentence embedding techniques, for instance, SkipThought vectors [15], FastSent [16], convolution neural networks (CNN) [17], recurrent neural networks (RNN) [18], [19].

BERT, or Bidirectional Encoder Representations from Transformers, is a method of pre-training language representations that obtains state-of-the-art results on many tasks, such as question answering and natural language inference. In this work, we leverage LaBSE (Language-Agnostic BERT Sentence Embedding) model to convert address strings to embedding vectors.

We use the output embedding of the LaBSE model [20] for each raw address input, which is a 768-dimensional vector. For records in the RAT table, we first map each record into a string sentence by concatenating $non-null$ values in the order from left to right, then feed this string sentence into the same LaBSE model to receive its corresponding 768-dimensional vector. Generally, LaBSE helps us reducing trainable parameters in our proposed model.

3) CRF-based token filtering: As discussed in , our RAT table only consists of street, ward, district, and city attributes. However, raw address inputs may contain irrelevant entities (e.g., project name, house number, etc.), punctuation and unrelated words outside the address scope. Thus, it is reasonable to filter unnecessary tokens before feeding them into our Siamese-based model.

We use a Name Entities Recognition (NER) model, more precisely, a Conditional Random Fields (CRF) model for this task. We clean the raw address input by keeping the sub-string part, which is bounded from the Street tag to the City tag if these tags exist.

Our pseudo code is as follows:

Algorithm 1: CRF-based token filtering

Result: Cleaned address string
initialization;
tags = NER(raw_address_string);
start_index = 0;
end_index = len(raw_address_string);
if *STREET in tags* **then**
  start_index = tags[STREET].left_index;
end
if *CITY in tags* **then**
  end_index = tags[CITY].right_index;
end
cleaned_address_ string =
 raw_address_string[start_index:end_index];

Figure 6 shows an example of token filtering output.

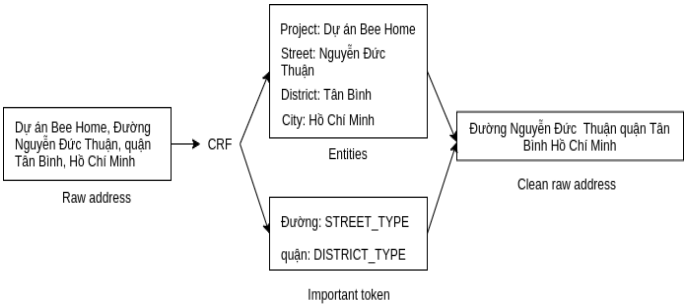

Figure 6: Example of token filtering task

4) Usage of token type feature: LaBSE model can encode an arbitrary string into a 768-dimensional vector. LaBSE does not take into account the particular characteristics of an address string. To facilitate model learning, we experiment with the additional usage of token type information in the feature engineering block. Token type features are 4-bits vectors f, which represents the existence of specific fields: $street$, $ward$, $district$, $city$, in the address string.

For a standardized address in the RAT table, the construction of the associated token type feature is obvious. Only the associated bit fields for non_null values are set to 1.

For raw address input, we leverage the output of the CRF tagging model discussed in Section III-C3. If one of these entities: $\{street, ward, district, city\}$ are detected, the corresponding bit in the token type vector f is assigned to 1. Otherwise, its bit will be zero.

We illustrate the creation of v_{raw} and v_{std} in Figure 7 and Figure 8.

IV. Evaluation

A. Datasets

Experiments are conducted on our crawled dataset from multiple sources such as real-estate, e-commerce, and social network websites. Raw addresses were labeled by hand. We have 41165 samples in total: a slip of 20% is reversed for a test set and the rest are used to train model. Our dataset is 2.5 times bigger than the previous dataset that was presented in [1]. The details of dataset is shown in Table III.

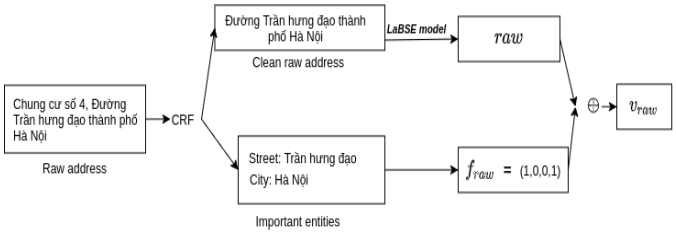

Figure 7: Example of creation of raw address vector

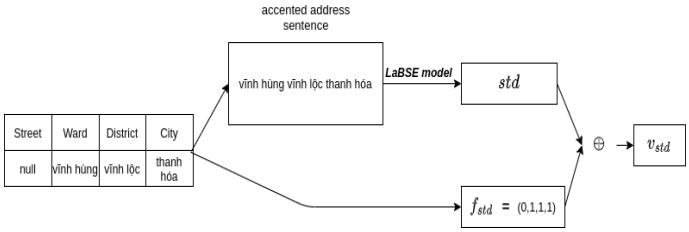

Figure 8: Example of creation of standard address vector

Dataset	Train	Test
Crawled dataset	32932	8233

Table III: Dataset

B. Experiments

Our network architecture is implemented using the Tensorflow framework. In the embedding block, we experiment with a multi-layer deep Siamese neural structure consisting of one input layer, two hidden layers, and one output layer. The input layer size equals 772, which is the size of 768-dimensional vector output from the LaBSE model plus the 4-dimensional vector of the token type vector. The size of hidden layers is set as 512, 256, respectively. The size of output layers is defined as 128.

Our ranking block consists of 4 hidden layers with the size of 64, 32, 16, 8 in the respective order. We use LeakyReLU (α = 0.1) as the activation function of the hidden layers for three reasons: First, LeakyReLU is cheap to compute since there is no complicated math, so the model can take less time to train or run. Second, it does not have the vanishing gradient problem led by other activation functions such as $sigmoid$ or $tanh$. Third, it is better than a ReLU function because it tends to avoid the "dying ReLU problem, " making the model fail to update parameters. The output layer of our ranking block has the size of 1, which outputs the similarity score. We use the $sigmoid$ activation function in the output layer.

In the training phase, we use batches of 1024 address pairs (each pair are a raw address and a standardized address). Negative samples are prepared by randomly picking a standardized

Method	Accuracy
NER model	88.31%
Stavia model	89.45%
Merge model	92.40%
Element-Wise model	94.14%
Absolute-Difference model	**95.41%**
Addition model	92.48%

Table IV: Model accuracy evaluation

address for each raw input. To optimize the network, Adam optimizer is employed with initial learning rate = 0.0001, β_1 = 0.9, β_2 = 0.999.

We train our models for 100 epochs. The training time for each epoch is about 20 minutes.

We evaluate our neural network architecture with four variants in the ranking block: Merge, Element-Wise, Absolute-Difference, and Addition models. We compare our models with two common approaches: a NER-based model and the Stavia [1] model. We report $accuracy$ scores: the number of true samples (x, y) (x is a raw input and y is a corresponding standardized address) divided by the number of samples in the test set, as the main metric.

Table IV shows the results of the experiments. Our best model, Absolute-Difference in the ranking block, obtains the highest score than the two previous approaches with 95.41% accuracy. The accuracy of the NER model depends heavily on the CRF model's performance, which is dependant on the dataset coverage. The Stavia model gets a slight improvement thanks to the added fuzzy-based ranking block.

V. Conclusions

In this paper, we propose a deep learning-based approach to address standardization challenges. Our key idea is to leverage Siamese neural network model to embed raw inputs and standardized addresses into a single latent multi-dimensional space. The embedding vector of the raw input will be used to rank over the embedding vectors of the existing standardized addresses in a RAT table. Thus, the corresponding output is the one with the highest-ranking score. In addition to the four variant models, we design several feature engineering tactics to boost the model performance.

We have collected and labeled 41165 samples, which is 2.5 times bigger than the previous public dataset. Our experiments demonstrate that our best model achieved 95.41% accuracy, which is significantly better than the latest SOTA. Besides performance, our approach is also superior in terms of execution time. In the inference phase, the embedding vectors of the standardized addresses can be cached. The ranking can be done in batch as typical execution in deep learning frameworks.

In the future, we plan to demonstrate our approach to different languages. We also investigate a bigger dataset at a more detailed level, including fields such as house number, alley, and project name.

References

[1] H.-N. Bui and V.-T. Tran, "A novel conditional random fields aided fuzzy matching in vietnamese address standardization," in *Proceedings of the Tenth International Symposium on Information and Communication Technology*, 2019, pp. 23–28.

[2] J. Buckley, B. Buckles, and F. Petry, "Processing noisy structured textual data using a fuzzy matching approach: Application to postal address errors," *Soft Comput.*, vol. 4, pp. 195–205, 12 2000.

[3] Y. Li, Q. Zhao, and Y. Yan, "Fuzzy matching of semantic class in chinese spoken language understanding," *IEICE Transactions*, vol. 96-D, pp. 1845–1852, 2013.

[4] A. Chatterjee, J. Anjaria, S. Roy, A. Ganguli, and K. Seal, "Sagel: Smart address geocoding engine for supply-chain logistics," in *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. SIGSPACIAL '16. New York, NY, USA: ACM, 2016, pp. 42:1–42:10. [Online]. Available: http://doi.acm.org/10.1145/2996913.2996917

[5] V. Borkar, K. Deshmukh, and S. Sarawagi, "Automatic segmentation of text into structured records," *ACM SIGMOD Record*, vol. 30, 01 2003.

[6] X. Li, H. Kardes, X. Wang, and A. Sun, "Hmm-based address parsing with massive synthetic training data generation," in *Proceedings of the 4th International Workshop on Location and the Web*, ser. LocWeb '14. New York, NY, USA: ACM, 2014, pp. 33–36. [Online]. Available: http://doi.acm.org/10.1145/2663713.2664430

[7] T. Churches, P. Christen, K. Lim, and J. Xi Zhu, "Preparation of name and address data for record linkage using hidden markov models tim churches," *BMC Med Inform Decis Mak*, vol. 16, 07 2004.

[8] P. Christen, T. Churches, A. Willmore *et al.*, "A probabilistic geocoding system based on a national address file," in *Proceedings of the 3rd Australasian Data Mining Conference, Cairns*, 2004.

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

[10] S. Sharma, R. Ratti, I. Arora, A. Solanki, and G. Bhatt, "Automated parsing of geographical addresses: A multilayer feedforward neural network based approach," in *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, Jan 2018, pp. 123–130.

[11] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," *Advances in neural information processing systems*, pp. 737–737, 1994.

[12] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2. Lille, 2015.

[13] I. Melekhov, J. Kannala, and E. Rahtu, "Siamese network features for image matching," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 378–383.

[14] R. Vinayakumar and K. Soman, "Siamese neural network architecture for homoglyph attacks detection," *ICT Express*, vol. 6, no. 1, pp. 16–19, 2020.

[15] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler, "Skip-thought vectors," *arXiv preprint arXiv:1506.06726*, 2015.

[16] F. Hill, K. Cho, A. Korhonen, and Y. Bengio, "Learning to understand phrases by embedding the dictionary," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 17–30, 2016.

[17] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

[18] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," *arXiv preprint arXiv:1705.02364*, 2017.

[19] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," *arXiv preprint arXiv:1508.05326*, 2015.

[20] F. Feng, Y. Yang, D. Cer, N. Arivazhagan, and W. Wang, "Language-agnostic bert sentence embedding," *arXiv preprint arXiv:2007.01852*, 2020.