Duplicate Identification Algorithms in SaaS Platforms

Dac Nguyen
AISIA Research Lab
Ho Chi Minh, Vietnam

Quy H. Nguyen
AISIA Research Lab
Ho Chi Minh, Vietnam

Minh-Son Dao
National Institute of Information and Communications Technology
Tokyo, Japan

Duc-Tien Dang-Nguyen*
Department of Information Science and Media Studies
University of Bergen
Bergen, Norway

Cathal Gurrin
Dublin City University
Dublin, Ireland

Binh T. Nguyen†
AISIA Research Lab
VNU HCM - University of Science
Ho Chi Minh, Vietnam

ABSTRACT

Existing duplicate records is one of the most common issues in many Software-as-as-Service (SaaS) platforms. In this paper, we study the duplicate identification problem in one specific SaaS platform related to quality and compliance management by using the address information. We interpret all typical mistakes from users that can generate the existent duplicated organizations in a given dataset, collected from the SaaS platform. Also, we create another set by crawling location data from Open Address (US Zone). We compare different methods, including Bag-of-words (using Cosine Distance), Record Linkage Toolkits, and Siamese Neural Networks using the triplet loss, in terms of precision, recall, and F1-score. The experimental results show that using Siamese Neural Networks can achieve a better performance in comparison with other techniques. We plan to publish our Open Address dataset and all implementation codes to facilitate further research in the related fields.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning algorithms**; **Neural networks**; **Classification and regression trees**.

KEYWORDS

Duplicate Identification, Siamese, Triplet Loss, bi-GRU, Software-as-a-Service

ACM Reference Format:
Dac Nguyen, Quy H. Nguyen, Minh-Son Dao, Duc-Tien Dang-Nguyen, Cathal Gurrin, and Binh T. Nguyen. 2020. Duplicate Identification Algorithms in SaaS Platforms. In *Proceedings of the 2020 Intelligent Cross-Data Analysis and Retrieval Workshop (ICDAR '20), June 8, 2020, Dublin, Ireland*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3379174.3392319

*Senior author
†Corresponding Author

ICDAR '20, June 8, 2020, Dublin, Ireland

ACM ISBN 978-1-4503-7509-2/20/06…$15.00
https://doi.org/10.1145/3379174.3392319

1 INTRODUCTION

Nowadays, there have been more and more Software-as-a-Service (SaaS) platforms in different aspects of daily lives in the fourth industrial revolution. People focus on the digital transformation of industry through connected systems, applications, and the gathering and harnessing of useful data from customers to foster their businesses. During the last decade, machine learning, data analytics, cloud services, and Software-as-a-Service (SaaS) are being integrated into businesses of all sizes to create intelligent networks and provide valuable data. As a result, SaaS platforms continue growing rapidly due to its ability to integrate with a third-party solution smoothly in an Industry 4.0 connected world. Efficient data aggregation and storage from different data sources become necessary for many SaaS products. Due to the blooming of data and data mining tools in recent years, each company has to choose smarter ways to restructure their data into a managed manner. Noticeably, data's characteristics depend on the corresponding product, the architecture, and the final implementation of the company. Consequently, data can vary much on size, shape, structure, and concept.

People habitually use the mail addresses as the essential information for identifying customers in SaaS platforms. The insertion process can be done manually and sometimes create multiple duplicate records due to human mistakes or without checking existent users. Typically, two addresses can be considered as a duplication when they represent a unique entity, located at the same place on the map. It is worth noting that these addresses usually contain many misleading details, dialects, and acronyms. Also, the structure of one address may be various among many countries. One may break these rules and write the addresses down in a different way. It creates more challenges to identify duplicated addresses accurately. Consequently, it is critical to investigate an efficient duplicate identification algorithm for each SaaS company to have a better way of managing new data coming and avoiding any duplication as much as possible. It can help to keep the stability and integrity of the customer data.

There have been multiple studies during the last two decades. Gao and colleagues [3] present a duplicate detection algorithm for light blogs and short comments by using the Word2Vec method and Hamming distances. They compare their proposed methods with the unweighted Word2vec method and the traditional TF-IDF technique on the SICK corpus. The experimental results show that using weighted Word2Vec can achieve higher accuracy and

recall rate than other techniques. Mudgal and co-workers [10] study a deep learning approach for entity matching and show that using deep learning approaches does not outperform the current solutions on structured entity matching. Still, it can significantly outperform them on textual and dirty entity matching with interesting experimental results. Hajishirzi et al. consider a novel near-duplicate document detection method by representing each document as a real-valued sparse K-gram vector and then optimizing the corresponding weights for a specified similarity function. The experimental results on two datasets (Web news articles and email messages) show that the new approach can outperform with the commonly used methods. More related works can be found in [4, 5, 8, 12].

Deep learning techniques nowadays become state-of-the-art methods in solving different problems, especially for computer vision and natural language processing fields. There have existed a huge number of useful applications using deep learning techniques in different aspects of daily life. Deep neural networks can also be applied to brain segmentation [11], automatic music generation [1], image recognition [9], entity matching [10], and question deduplicating [4, 5].

In this paper, we investigate the duplicate identification problem in one SaaS company related to quality and compliance management. The company provides us one original dataset which has 5600 existent organizations (including both duplicated and non-duplicated cases). We aim at studying the duplicate identification problem by using the address information based on this dataset. Besides, we take another approach by analyzing all common mistakes from users that can generate the existent duplicated organizations in the dataset given. By crawling location data from Open Address[1] (US Zone) and making new samples by different types of users' mistakes learned, we create the second dataset for extensive experiments. We apply different techniques, including Bag-of-words (using Cosine Distance), Record Linkage Toolkits, hybrid methods, and Siamese Neural Networks using the Triplet loss, for comparing the corresponding performance in terms of precision, recall, and F1-score, to choose the most appropriate model. The experimental results show that using Siamese Neural Networks can achieve a better performance than other techniques in both datasets.

2 METHODOLOGY

In this section, we present our methods to detect duplicate items by using addresses. As the dataset provided by the SaaS company having confidential information from customers, we will not disclose it in details but examine all possible human mistakes related to the existent duplications. Subsequently, we demonstrate the performance of different learning models via another dataset, crawled from the website Open Address.

2.0.1 Preliminary. Given two strings, both of which represent two mail addresses and point to a unique physical identity that can be determined as a duplicate address. In what is going, we first discuss how we process and analyze all duplicate data based on learning patents from users' behaviors in the dataset provided by the SaaS company. Then, we describe all learning models to solve the current problem. For the sake of easiness, we denote dataset A as the dataset provided by the SaaS company and dataset B as the dataset generated by crawling addresses from the website Open Address.

[1] https://openaddresses.io/

2.1 Duplication Analytics

There are 5600 organizations in the dataset A, where 2.34% of the number of organizations is determined as duplicated items. As one organization may have more than one duplication, the total number of records related to these organizations is 5787. Understanding deeply all existent cases related to duplication issues in the dataset A can give us another chance to find appropriate features for model learning.

It is worth noting that all factory addresses in the dataset A are written in English. Commonly, an address can have multi-components such as address number, street name, postal code, and place name. These sub-components do not need to be available at the same time or the same resolution or follow a precise rule as the writing style influences heavily. For instance, one can consider the following example. A Vietnamese may prefer using "277 Nguyen Van Cu, District 5, HCMC, Vietnam" instead of "#277, Nguyen Van Cu, Ward 4, D5, HCM, Vietnam". Here, two street addresses are identical in terms of the location on the map, but the second one gives more details on a focused area (Ward 4) while using short terms of both district and city. Besides, one can find the different structure in this example; the address name may go after the symbol # and separate its following term by a comma.

In the dataset A, all existent users come from 77 countries with different geographical hierarchy; hence it creates more challenges for the duplicate identification problem. To demonstrate those challenges, we list all possible levels that can occur in addresses across four countries (Vietnam, U.S.A, South Korea, and Ecuador) obtained from the dataset A and depict them in Table 2. Since every country has a different geographical hierarchy for the address format due to its culture, we only use three values, YES, NO, and OPTION for answers in each level. If a level has a "YES", it means it is a reliable indicator such that if the address does not include it, people hardly find the correct location. For instance, some cities in South Korea have one district, namely Buk-gu. If the province or city name is not provided, it is impossible to identify where the address is. The "NO" value indicates that the corresponding level neither exists in an official hierarchy nor the associated language. Finally, "OPTIONAL" means the address may (not) need the information of this level without making any difficulty in determining the exact location.

Table 1 depicts all popular reasons generating duplicate organizations in the dataset A. The spelling mistake is a predominant case where users may incorrectly type some words in a given address. Wrong wording is a respectable minority in which users sometimes use incorrect words for typing the corresponding addresses to refer to a given organization. Due to the immature stage of the SaaS platform, system errors can create "Null" values and produce duplicate records later. The writing style is vast various with synonyms and acronyms, very often used by our users when they want to complete filing the organization address quickly. We categorize those causes that do not have a high frequency in the last group. Figure

1 illustrates the distribution of spelling errors, wrong wordings, system errors, synonyms, acronyms, and others in the first dataset.

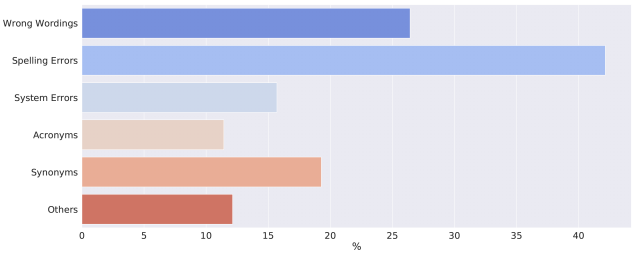

Figure 1: The distribution of common reasons creating duplicated organizations in the SaaS platform. The most popular reason comes from spelling errors (42.14%) of users.

2.2 Data Augmentation

For every machine learning model, training data are crucial for success. It is important to note that in the SaaS dataset provided, the number of samples is relatively small. Also, understanding all possible mistakes from individuals and systems that generate duplicated items can give a better way to reproduce, extract essential features, and learn suitable machine learning models for the problem. For a given address, we apply the following preprocessing and augmentation steps:

(1) We lower case all characters.
(2) We remove all punctuation marks.
(3) We remove all simple mistakes (by human or system) such as, e.g., multiple spaces, characters "Null" or "'NaN".
(4) Following the results of the duplication analytics, we create a heuristic rule to replicate human/system mistakes for generating more samples in the training dataset.
(5) Finally, for diversifying duplicate cases in the experiments, we use further techniques (including random insertion, swap, and deletion) by using a Python library, namely **nlpaug**[2].

2.3 Model Learning

The essential substance of any duplicate identification approach is based on a robust metric. In literature, there are several token-based metrics for this purpose. Jaro distance is the weighted sum of the percentage of matched characters, while Jaro Winkler improves this measure for matching prefix characters [14]. Also, Levenshtein and Damerau Levenshtein determine the distance as the minimum number of single-character edits (such as, e.g., insertion, deletion, substitution) [7, 8]. Moreover, Q-Gram approximates the string-matching score by finding common Q-grams, which are substrings of length Q [13]. Longest Common Subsequence (LCS) finds the longest common subsequence in two strings or sequences [7, 8]. Despite their usefulness, they may not well distinguish two inputs that do not have a similar structure regardless of the same content due to natural language characteristics. To expand the capability of the string distance, we use the Siamese network for constructing a deep learning duplicate prediction model for the problem.

2.3.1 The Siamese network [6]. It is an instance of deep metric learning, which can estimate the similarity between two inputs. Its underlying design consists of shared parameters for children networks. Together, sub-networks project the input data onto an abstract dimension in which the difference computed can be maximized as much as possible. Typically, one can use the Siamese network's characteristics to imply whether two input data are matching or not, under one threshold. The architecture of the Siamese neural network used in our experiments can be depicted in Figure 2.

2.3.2 Triplet Loss. It is a revisiting approach to enhance Siamese networks. Traditionally, the network consumes positive pairs, neglects negative pairs. A recent study [2] explains that using both of them can give more reliable feedback for back-propagation, and then lead to a robust result. Figure 2 shows the architecture of training a Siamese model using the triplet loss. Given a candidate, one can find both positive and negative samples by collating associated labels, belonging to a similar one, which is positive, contrarily negative. We first feed three samples through our network at once, then compute the negative and positive distance. The process ends at the triplet layer, top layer, where the network minimizes the loss functions.

The loss function can be defined as follows:

$$\mathcal{L} = \sum_{i=1}^{N} \text{ReLU}(\| \text{f}(x_i^a), \text{f}(x_i^p)\| - \| \text{f}(x_i^a), \text{f}(x_i^n)\| + \alpha), \qquad (1)$$

where x^a, x^p, and x^n denote anchor, positive, and negative, respectively. The notation $\text{f}(\cdot)$ represents for the corresponding result of the network transformation layers, α is a margin, and $\|\cdot, \cdot\|$ is an arbitrary distance. We describe our selection of those distances at Table 3.

Each of our proposed children networks consists of three layers; they are embedding, bi-GRU, and the linear readout layer (Figure 2). Subsequently, a triplet loss stands on the top, aggregates negative and positive pairs. First, all input strings are padded to produce a sequence of M characters. Secondly, at the embedding layer, we use the character-based embedding to transform each character in an address into a vector whose size is H. Next, the bi-directional GRU processes the sequence of vectors one by one in two ways, one from the first character to the last and vice versa. During this processing phase, for every time step in each direction, the previous hidden state will be taken. At each time step, we extract the hidden state from the forward and backward networks and concatenate them. Hence, the bi-GRU's shape is $M \times (2 * H)$. Lastly, the linear readout layer squeezes each concatenation to a fixed length of L by a linear transform; later, a sum-pooling is used to obtain the final output $1 \times L$.

3 EXPERIMENTS

In this paper, we execute all experiments on a computer with Intel(R) Core(TM) i7 2 CPUs running at 2.4GHz with 128GB of RAM and an Nvidia GeForce RTX 2080Ti GPU. We measure the performance of different methods by using the Record Linkage Toolkit[3] (RLT),

[2]https://github.com/makcedward/nlpaug

[3]https://recordlinkage.readthedocs.io/en/latest/about.html

Type	Expectation	Variant
Spelling errors	No 8 **Mangu** Road, Beijiao, Shunde	No. 8 Manggu rd., Beijao, Shune
Wrong wordings	82 **Muhak-ro**, Yongdu-dong, Seoul	82 **Muhak ro**, Yongdu dong, Seoul
System errors	59 Hanoi highway, Thao Dien, D2, HCM, **760000**	59 Hanoi highway, Thao Dien, D2, HCM, **null**
Synonyms	Economic development **zone**	Economic development **area**
Acronyms	277 Nguyen Van Cu **street**	277 Nguyen Van Cu **str**
Others	Lot 2, Road 13, Amata Industrial Zone, **Bien Hoa**, Dong Nai, VN	Lot 2, Rd. 13, Amata Industrial Zone, Dong Nai, VN

Table 1: The existent reasons generating duplicate items by users in the SaaS platform. There are six types of reasons, including spelling errors, wrong wordings, system errors, synonymns, acronyms, and others.

Resolution Level	Vietnam	U.S.A	South Korea	Ecuador
Other Indicators (e.g. "km. 1.5 via Duran Tambo")	Optional	No	No	Optional
Address Number	Optional	Yes	Yes	Optional
Street Name	Yes	Yes	Yes	Yes
Street Type	Optional	Yes	Optional	Optional
Occupancy Type	Optional	Optional	Optional	Optional
Industrial zone	Optional	No	No	No
Neighborhood	Optional (Hamlet, Village, Commune)	No	Yes (Dong)	No
Town	Yes	No	Optional	Yes
District	Yes	Optional	Yes	Optional
City/Province/State	Yes	Yes	Yes	Yes
Postcode	Optional	Yes	Yes	Optional

Table 2: The different geographical hierarchy of one address format in four countries (Vietnam, USA, South Korea, and Ecuador). Each resolution level has three values: Optional, No, and Yes.

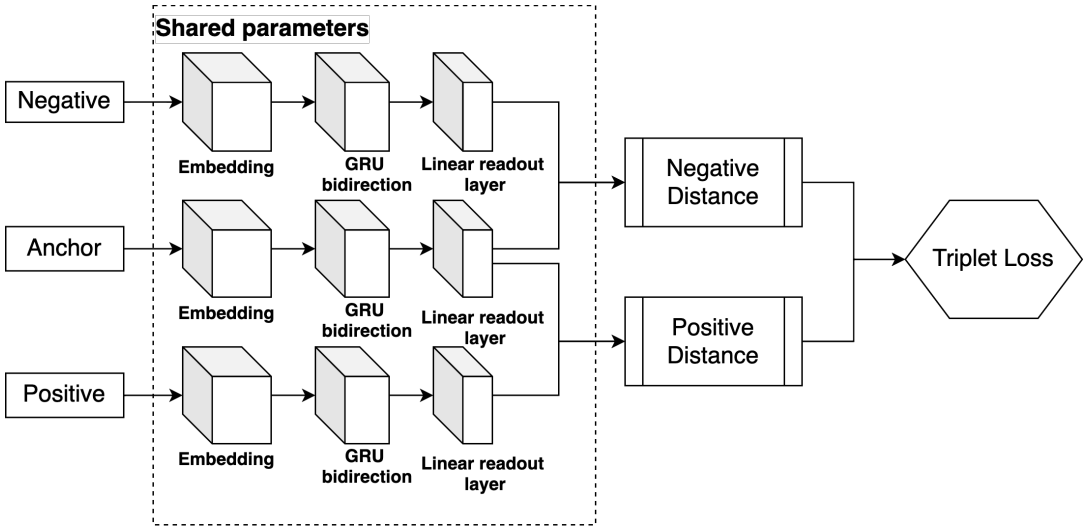

Figure 2: The architecture of our proposed Siamese model.

the proposed Triplet Siamese model, and choosing the following metrics: precision, recall, and F1-score.

3.1 Datasets

It is crucial to remark that a triplet Siamese model requires three inputs: anchor, positive, and negative, which can be consumed at once during training. The set has three types of data called a triplet, in which the anchor represents a unique address; the positive is the duplicate information; in contrast, the negative is a sample different from the anchor. Hence, it is reasonable to create a set of data that has three different types of samples.

In our experiments, we use two datasets, as mentioned previously. The first training dataset, dataset A, has 5600 unique organizations. Initially, 131 organizations have duplicate records, but 22 triplets

(a) and 109 positive pairs (b) are available. To populate (b), we randomly selected samples that are not duplicate records to become negative components. When having enough triplets, we apply our proposed augmentation techniques, as mentioned preceding section. Simultaneously, an additional effort to increase the number of samples is conducted. We use a characteristic learned from the analytics phase; that is, pairs have different street numbers (c). Thus, we follow this presentation property to find ten negative pairs. For each of the couples, we choose a candidate to be an anchor, and later augmentation is applied over it, yield a positive sample. The total number of samples when we finished this (a), (b), and (c) is 31424.

Next, we consider one Open Address dataset containing nearly 27 million addresses (US-EN only). Among these millions of addresses, we randomly collected about 2000 ones. It is important to note that no duplication exists here. Similarly, by applying our proposed augmentation techniques, we achieve the dataset B of 43978 locations. We regard 2000 selected addresses as anchors, use the corresponding augmented samples created for each address as positives, and select any of the remaining unique addresses as negatives.

3.2 Results

For measuring the performance of our proposed model and other techniques, we consider the following scheme of experiments. As mentioned in the previous section, we train the proposed Siamese model by studying two different training datasets: one using the dataset A (namely **Triplet Siamese (A)**) and another using the dataset B (namely **Triplet Siamese (B)**).

During the training process, we use one validation dataset including 49 positive pairs (anchor and positive) of duplicate addresses

$$\{(e_1, f_1), (e_2, f_2), \ldots, (e_{49}, f_{49}\}$$

collected from our platform. Here, each (e_i, f_i) maps to the same organization ($i = 1, \ldots, 49$). It is important to remark that these addresses are completely different from ones in training datasets. Subsequently, we add other 1176 negative pairs (anchor, negative) along with 49 positive pairs to obtain 1225 pairs for the validation dataset. Similarly, we consider other 114 positive pairs (anchor and positive) of duplicate addresses as well as 5000 negative pairs (anchor, negative) to generate the testing dataset.

To obtain the most suitable model in this problem, we compare different configurations of the proposed Siamese neural networks: GRU/LSTM + Cosine/Tanh, as described in Table 3. One can see that using GRU+Cosine can achieve the best performance in terms of F1-score. As a result, we decide choosing the final architecture as described in Figure 2.

In the experiment, we choose the embedding layer having 300 dimensions and bidirectional GRU having 120 hidden dimensions. The final output vector of the linear readout layer has 50 dimensions.

For seven different distance and similarity methods (Jaro Winkler distance, Levenshtein distance, Damerau Levenshtein distance, Q-Gram, Longest Common Subsequence (LCS), Smith-Waterman, and the cosine distance), we execute necessary preprocessing steps for the input data. Then, we select the best thresholds to classify two given addresses are matching or not in terms of the computed similarity score. To use the cosine distance, we first tokenize all the addresses into tokens; then, we apply the cosine similarity to measure the similarity between two addresses. Importantly, when tuning learning parameters and thresholds for each learning model or similarity methods, we consider different thresholds in $\{0.10, 0.11, \ldots, 0.98, 0.99\}$ and find the best one by measuring the corresponding performance in the validation dataset using the F1-score. Next, we apply these learning thresholds to measure the performance of each method in the testing dataset. Table 2 describes the best performance of different techniques in the validation dataset, where using the triplet Siamese model with the TS-Lab dataset outperforms all other classifiers in terms of all F1-score, precision, and recall. The best thresholds for all approaches (Triplet Siamese (A), Triplet Siamese (B), Jaro Winkler distance, Levenshtein distance, Damerau Levenshtein distance, Q-Gram, Longest Common Subsequence (LCS), Smith-Waterman, and the cosine distance) are 0.57, 0.72, 0.66, 0.33, 0.33, 0.37, 0.59, 0.28, 0.57, respectively.

Additionally, we study one hybrid approach by combining seven distance-based models and considering the predicted similarity of each distance-based model as input data of the hybrid model. In our experiments, we apply the logistic regression method for learning the hybrid model. The corresponding output is matching or not matching. It is worth noting that our proposed methods using both Triplet Siamese (A) and Triplet Siamese (B) outperform all other techniques in terms of the F1-score in the testing dataset. More detailed, using Triplet Siamese (A) achieves the highest F1-score (68.36%) while Triplet Siamese (B) gets 64.26% in F1-score. The Cosine and Smith Waterman distances can obtain the best recall (93.18%) while the hybrid model only achieves 90.91% in recall.

4 CONCLUSION

In this paper, we have addressed a duplicate identification problem that commonly happens in SaaS platforms. The problem's complexity increases concerning the cost of collecting data and various user behaviors across different countries. We have overcome the problem with an empirical study in which we analyze common errors that occur in the SaaS's production, which helps to reproduce these mistakes upon the training set, to enlarge training samples as well as making the machine learning model more robust. Besides, we have proposed an approach using a character-based triplet Siamese model to act as a deep learning metric, which computes the similarity between two given addresses. We measure the performance of different models in terms of accuracy, F1-score, precision, and recall. The experimental results have shown that the proposed methods outperform other traditional string metrics in terms of F1-score. Those results can bring many benefits to a SaaS platform, which reduces data fragmentation across different clients and ecosystems. In the future, we plan to continue improving our methods by applying new algorithms related to the problem, such as active learning, to leverage crowdsourcing to label new data.

ACKNOWLEDGEMENT

This research is conducted under the Collaborative Research Agreement between National Institute of Information and Communications Technology and University of Science, Vietnam National

	LSTM+Cosine	LSTM+Tanh	**GRU+Cosine**	GRU+Tanh
F1-score	0.82	0.7788	**0.8396**	0.7463

Table 3: The comparison among different configurations of the proposed Siamese models: LSTM + Cosine, LSTM + Tanh, GRU + Cosine, and GRU + Tanh.

	Accuracy	F1	Precision	Recall
Triplet Siamese (A)	**0.9736**	**0.8396**	**0.9082**	0.7807
Triplet Siamese (B)	0.9527	0.5323	0.44	0.6735
Jaro Winkler	0.904	0.4633	0.3083	0.9318
Levenshtein	0.9333	0.56	0.3962	0.9545
Damerau Levenshtein	0.9323	0.5563	0.3925	0.9545
Q-Gram	0.9202	0.5212	0.3554	0.9773
Cosine	0.9444	0.5985	0.4409	0.9318
Smith Waterman	0.9576	0.6769	0.5116	**1.0**
LCS	0.9556	0.6562	0.5	0.9545
Logistic Regression	0.9313	0.5641	0.3929	**1.0**

Table 4: The performance on the validation set of seven distance-related methods, the ensemble logistic model, and the triplet Siamese model trained by two training sets - (A) is the dataset A, (B) is the dataset B.

	Accuracy	F1	Precision	Recall
Triplet Siamese (A)	**0.983**	**0.6836**	**0.5839**	0.8246
Triplet Siamese (B)	0.9806	0.6426	0.546	0.7807
Jaro Winkler	0.9587	0.2749	0.1619	0.9091
Levenshtein	0.9601	0.2817	0.1667	0.9091
Damerau Levenshtein	0.9595	0.2787	0.1646	0.9091
Q-Gram	0.9758	0.3922	0.25	0.9091
Cosine	0.9671	0.328	0.199	**0.9318**
Smith Waterman	0.9693	0.3431	0.2103	**0.9318**
LCS	0.9787	0.4233	0.2759	0.9091
Logistic Regression	0.9671	0.3226	0.1961	0.9091

Table 5: The performance on the testing set of seven distance-related methods, he ensemble logistic model, and the triplet Siamese model trained by two training sets - (A) is the dataset A, (B) is the dataset B.

University at Ho Chi Minh City. We acknowledge the support of Science Foundation Ireland under grant number SFI/13/RC/2106 and L. Meltzers Høyskolefonds, UiB 2019/2259-NILSO.

REFERENCES

[1] Han K. Cao, Duyen T. Ly, Duy M. Nguyen, and Binh T. Nguyen. 2019. Automatically Generate Hymns Using Variational Attention Models. In *Advances in Neural Networks – ISNN 2019*, Huchuan Lu, Huajin Tang, and Zhanshan Wang (Eds.). Springer International Publishing, Cham, 317–327.

[2] Xingping Dong and Jianbing Shen. 2018. Triplet loss in siamese network for object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 459–474.

[3] J. Gao, Y. He, X. Zhang, and Y. Xia. 2017. Duplicate short text detection based on Word2vec. In *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. 33–37. https://doi.org/10.1109/ICSESS.2017.8342858

[4] Yukiko Homma. 2017. Detecting Duplicate Questions with Deep Learning.

[5] Doris Hoogeveen, Andrew Bennett, Yitong Li, Karin M. Verspoor, and Timothy Baldwin. 2018. Detecting Misflagged Duplicate Questions in Community Question-Answering Archives. In *ICWSM*.

[6] Bromley Jane, Guyon Isabelle, LeCun Yann, Säckinger Eduard, and Shah Roopak. 1993. Signature Verification Using a "Siamese" Time Delay Neural Network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems (NIPS'93)*. Morgan Kaufmann Publishers Inc., San Francisco CA USA, 737–744. http://dl.acm.org/citation.cfm?id=2987189.2987282

[7] Matthew A. Jaro. 1976. *UNIMATCH: A Record Linkage System: User's Manual.* Technical Report. U.S. Bureau of the Census, Washington, D.C.

[8] Matthew A. Jaro. 1989. Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. *J. Amer. Statist. Assoc.* 84, 406 (1989), 414–420. https://doi.org/10.1080/01621459.1989.10478785 arXiv:https://www.tandfonline.com/doi/pdf/10.1080/01621459.1989.10478785

[9] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese Neural Networks for One-shot Image Recognition.

[10] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep Learning for Entity Matching: A Design Space Exploration. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD '18)*. ACM, New York, NY, USA, 19–34. https://doi.org/10.1145/3183713.3196926

[11] D. M. H. Nguyen, H. T. Vu, H. Q. Ung, and B. T. Nguyen. 2017. 3D-Brain Segmentation Using Deep Neural Network and Gaussian Mixture Model. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 815–824.

[12] Thorsten Papenbrock, Arvid Heise, and Felix Naumann. 2015. Progressive Duplicate Detection. *Knowledge and Data Engineering, IEEE Transactions on* 27 (05 2015), 1316–1329. https://doi.org/10.1109/TKDE.2014.2359666

[13] Esko Ukkonen. 1992. Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science* 92, 1 (1992), 191 – 211. https://doi.org/10.1016/0304-3975(92)90143-4

[14] William E. Winkler and Yves Thibaudeau. 1991. *An Application of the Fellegi-Sunter Model of Record Linkage to the 1990 U.S. Decennial Census.* Technical Report Statistical Research Report Series RR91/09. U.S. Bureau of the Census, Washington, D.C.