GeoDD: End-to-End Spatial Data De-duplication System

Mykola Trokhymovych[1] and Oleksandr Kosovan[2(✉)]

[1] Ukrainian Catholic University, Lviv, Ukraine
trokhymovych@ucu.edu.ua
[2] University of Lviv, Lviv, Ukraine
Oleksandr.Kosovan.AEKE@lnu.edu.ua

Abstract. People generate vast amounts of data that can be used for analytics, data-driven decision-making, and forecasting. However, to extract value from data, we need to apply specific methods of cleaning and prepossessing it. In this paper, we observe the problem of geospatial data de-duplication, propose and implement end-to-end solutions for social-media-based data de-duplication. We apply advanced geospatial, natural language processing, and classical machine learning methods for our solution. Our tool shows high competitiveness in observed competition and can process a vast amount of data with limited computational resources.

Keywords: De-duplication · Spatial data · End-to-end system · Search · NLP

1 Introduction

The rise of social media platforms led to huge data flow generated by millions of people. It resulted in numerous tasks to allow analyzing and using that information. One of these tasks is data cleaning and de-duplication. In this work, we observe the task of spatial data de-duplication, on the example of Foursquare data provided within Kaggle competition.[1] The efficient and successful matching of spatial points helps to make it easier to identify where new stores or businesses would benefit people the most.

1.1 Problem Definition

Users can generate or complement information about places on the platforms. Also, data can be scraped or imported from other online sources. As a result, many places have duplicates described as exact locations. This work aims to

[1] Kaggle. Foursquare—Location Matching, https://www.kaggle.com/competitions/foursquare-location-matching.

R. Silhavy et al. (Eds.): CoMeSySo 2022, LNNS 597, pp. 717–727, 2023.
https://doi.org/10.1007/978-3-031-21438-7_60

improve place data quality by records de-duplication. In the real world, user-generated data can have mistakes, typos, and missing values. Another problem is that usually, we have a massive amount of records, which is impossible to compare by the brute force one-to-all approach, as to $O(n^2)$ complexity in the general formulation.

In this work, we present the solution to the de-duplication problem, that can efficiently work with huge amounts of data with affordable processing time and computational resources.

1.2 Related Work

Spatial data de-duplication problem is related to several tasks that involve machine learning and other data processing techniques, especially in the domains of search, NLP, and spatial data process.

Bohannon P. et al. considered the place duplicates problem, where a place contains a name and a physical location. Due to the lack of additional information about locations, the task was challenging. To resolve this problem, the authors use both domain approaches—NLP and spatial techniques. Also, the authors noted the difficulty of de-duplication for different place types, such as a shopping mall or a park that contain a different number of places [2].

Other work observes building unified embeddings for geopoints for search and de-duplication [9]. It also present the solution to make de-duplication of data using NLP and geospatial analysis, and highly inspire our solution.

Another research takes the place graph from Facebook to improve place de-duplication. In this data set, some fields contain additional place attributes like names, addresses, images, etc. In addition to NLP and spatial processing techniques, the authors used a case from the computer vision domain—person re-identification [9].

The authors detect duplicates in two steps: *(i)* generating pairs using blocking and *(ii)* classifying candidates by PlacERN. In turn, pair generation was divided into two fazes: *(i)* a partitioning phase: generating all possible pairs in place sets; *(ii)* a filtering phase: computing a Jaro-Winkler similarity metric for each pair with the defined threshold to avoid hurting recall [1].

Taking into account previous research results, we came out with our solution to this specific task.

2 Methods

In this section, we observe the methodology used for building an end-to-end spatial data de-duplicate system. We define the group of points corresponding to one entity as a Point of Interest (POI). The task is to group independent POIs and find duplicates among a massive set of 1M+ records.

We want our solution to be both accurate and efficient. So, we set time and resource limits to be 9 h of inference time and 12 GB of RAM.

2.1 Exploratory Data Analysis

The initial step of any data science project is data observation. For our research, we use the data from the Kaggle competition mentioned in the Introduction Sect. 1. Each entry of data includes attributes like the name, street address, coordinates, country, state, and category. The dataset consists of 1,138,812 records labeled 739,972 unique POIs, and 221 unique country labels. The final testing is performed on the hidden test part, to avoid over-fitting and data leakage.

We noticed that missing features for different columns and their rate differs. For example, we had columns that had almost always some value (latitude, longitude, name, country), and some columns were in $\geq 50\%$ records empty (URL, phone, zip). Rate of missing values per column presented in Fig. 1.

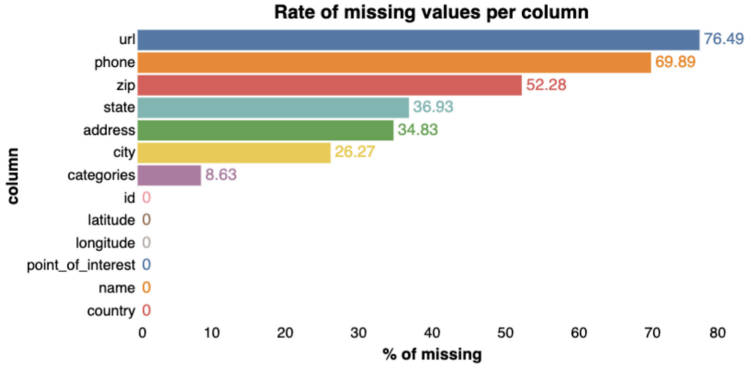

Fig. 1. Rate of missing values per column for train part of dataset

Also, we had different numbers of records for different countries. That influences metrics, as different countries have their specifics, resulting in different metrics scales for each. A number of records for each country have a different influence on the final results (Fig. 2).

Also, there was a tendency for different places in the world to have different densities of points. That also has a strong influence on the final prediction. For example, we observe big cities have a high density of points, the same works for coastlines or places of interest. Some anomalies were observed, such as in India, where all points are concentrated in particular locations. At the same time, Japan, and the UK, for example, have different situations, and points are equally distributed on the whole country's territory (Fig. 3).

The EDA presented in this section resulted in our specific approach to problem-solving.

2.2 General System Overview

We have designed our system as a two-stage model (Fig. 4). The first level was responsible for collecting the candidates to be a duplicate for each record.

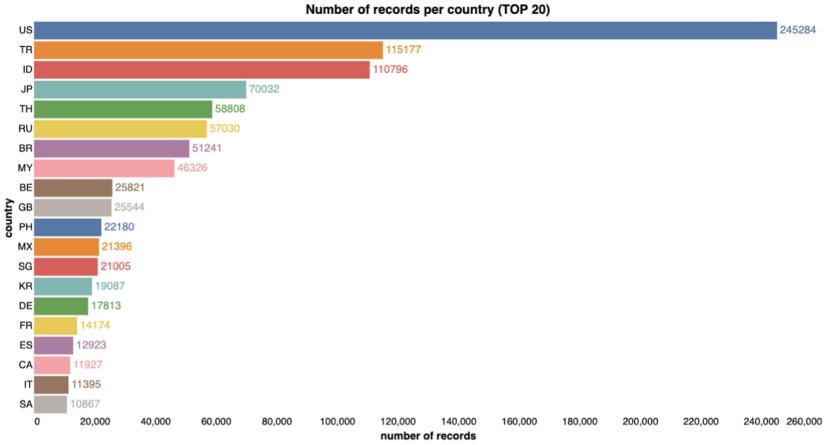

Fig. 2. Number of records per country (Top-20)

Fig. 3. Distribution of points across the globe

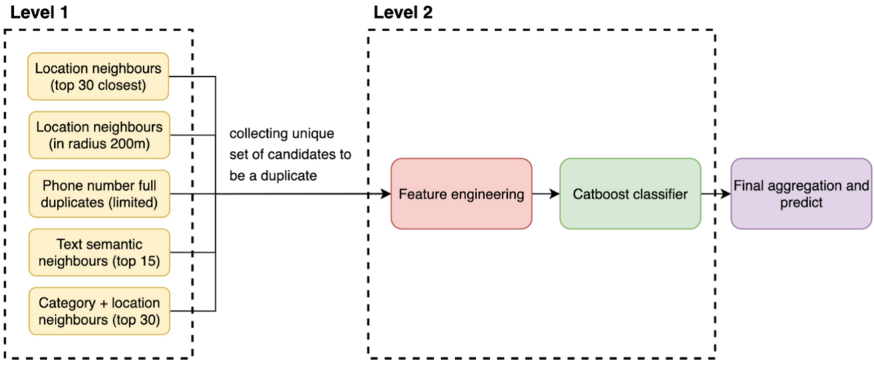

Fig. 4. End-to-end spatial data de-duplicate system schema

The second one was responsible for classifying real duplicates given the set of duplicates. The complete schema is presented in the following picture. During the initial EDA, we found that 99.87% of all pairs are from one country. As a result, we decided to process countries by groups (clusters). So we grouped countries into the following clusters: (TR, GR), (US, CA, MX), (JP), and (others).

It allowed us to reduce memory usage and increase processing speed as the indexes became smaller and didn't significantly change performance.

2.3 Stage One. Candidates Selection

As for the first level, we used several techniques for candidate selection.

The initial approach used only location coordinates (longitude and latitude). We used these features to create a BallTree index with haversine distance for efficient search of location neighbors [3]. We exploited it in two ways: finding the top-30 closest points to the given one and finding all points in the neighborhood of 200 m. These parameters were chosen based on an empirical study of our specific dataset to cover a maximum number of true duplicates without numerous candidates.

The second approach aimed to find the location neighbors with specific categories. We have trained a W2V model with existing categories [6]. We took all categories for specific POI, collected them to list, and treated each as a token in the sentence. Using such an approach, we could train embeddings of categories to ensure that similar categories correspond to close embeddings. We concatenated them with coordinate values to get the final vector to index. Finally, we used Non-Metric Space Library (NMSLIB) to build an approximate nearest neighbors (ANN) index and efficiently search for duplicate candidates [5].

The third approach was based on text semantic similarity. We used Sentence-Transformers library as a framework to fine-tune custom multilingual models [7]. As a base model, we took *paraphrase-multilingual-MiniLM-L12-v2* model as it works with multiple languages, shows good performance, and returns a vector of 384 dimensional, which is twice shorter than regular BERT or other similar models. We built a training dataset from the pairs obtained from the first two approaches. So we had pairs of places and their features. The label was binary, one when two places are from one POI and zero otherwise. We fine-tuned the model for the cosine similarity of the two texts. The text of place was constructed by joining all text features of place with the **[SEP]** token. So, the text for a specific place was constructed with the following pattern: *name **[SEP]** address **[SEP]** city **[SEP]** url **[SEP]** country **[SEP]***. Finally, we could make embeddings of texts of all places and index them using Non-Metric Space Library (NMSLIB) and cosine similarity distance measurement.

The last approach was to extract candidates using complete duplicates by phone number. We considered only numbers that occurred less than ten times to omit hotlines of big chains like McDonald's or IKEA.

2.4 Stage 2. Classification Model and Post-processing

Having selected candidates, we proceed with building a model to classify if the pair is a duplicate or not. For that, we decided to build an independent CatBoost binary classification model [4]. We used a simple model with default parameters, Log Loss, 25k iterations, and 0.01 learning_rate. Also, we used a 10% test split and best model selection based on metrics on the test. Results of our modeling will be presented with more details in Sect. 3.

The most computationally expensive part was feature engineering. We had an average of 81.07 candidates for each point in the dataset. We were processing all points in batches of 500k candidates to avoid going out of RAM limits.

We built and used features presented in Table 1, used in our classifiers.

After the final classification, we collected the pairs that were classified as positive pairs. We used two types of post-processing:

- If A is a pair of B, then B is a pair of A
- If A is a pair of B and B is a pair of C, then A is a pair of C

In that way, we extended the list of predicted pairs for each point, which resulted in better metrics.

3 Results

We evaluated each stage of the solution independently. For the first level, we used the recall metric, calculated with Eq. 1.

$$Recall = \frac{Number\_of\_real\_duplicates}{Number\_of\_items\_retrieved}. \tag{1}$$

As a result of the first stage model presented in Sect. 2.3, we achieved a 0.9786 recall for the first stage. More detailed results can be observed in Table 2. We could achieve such a recall only by combining different approaches to candidate selection, which independently had worse results. However, each of them covered their specific cases, which resulted in an excellent boost when combining them. We found out that different countries have their particular characteristics. For example, the average number of ground true candidates varies from 1.41 for CL to 6.62 for ID. The same works for potential duplicates, ranging from 59.19 for IN to 127.16 for SG. Taking this into account, we decided to build independent second-stage models for each country presented in Table 2.

As for the second-level model, we used different metrics to evaluate the model during its training. We used *Precision*, *Recall*, *F1* and *Accuracy* metrics in classical binary classification task formulation. Figure 5 shows the example of the performance of different models that were created for specific countries. Each CatBoost model trained 10 thousand iterations with a learning rate of 0.01 based on log loss.

Due to each country's specifics, we observe that metrics are varying for different countries.

Table 1. Features used for classification model

Feature names	Description
latitude_1, longitude_1, latitude_2, longitude_2	Coordinates of pair of points
category_encoding_1, category_encoding_2	Having a vector that represents the category, we applied the clustering technique to group all possible categories into 50 groups and used the id of the group
category_match_score	Having vectors representing each point's category in a pair, we calculate the cosine similarity between those vectors. In case we have more than one category for one point, we take a mean of category vectors as a vector representation of category for the specific point
semantic_match_score	Having vectors of text obtained with the logic presented in Sect. 2.3 for each point, we calculate cosine similarity between them
feature_inclusive	Categorical features correspond to the method or set of methods that returned this pair from the first level. For example, it shows that we got this pair from text similarity but not from categorical and location similarity
latdiff, londiff, manhattan, euclidean, haversine	Features that correspond to different methods of spatial distance calculation
name_levens, name_jaros, name_len_1, name_len_2, name_nlevens, address_levens, address_jaros, address_len_1, address_len_2, address_nlevens, city_levens, city_jaros, city_len_1, city_len_2, city_nlevens, state_levens, state_jaros, state_len_1, state_len_2, state_nlevens, zip_levens, zip_jaros, country_levens, country_jaros, url_levens, url_jaros, url_len_1, url_len_2, url_nlevens, phone_levens, phone_jaros, categories_levens, categories_jaros, categories_len_1, categories_len_2, categories_nlevens	Features correspond to Levenshtein, Jaro-Winkler char level distances between two texts, along with the length of those fields for name, state, city, phone, URL, categories, and address fields of point pairs

The final metric used to evaluate end-to-end solution is IoU. It shows how accurate we find the duplicates related to both finding the candidates and classifying them. It is calculated with Eq. 2.

$$IoU = \frac{P(\{predicted\_duplicates\} \cup \{true\_duplicates\})}{P(\{true\_duplicates\})}, \tag{2}$$

where P—is the power of the set function. As for a final prediction, we used the standard 0.5 thresholds for final prediction, as it showed the best IoU final result in our case.

Table 2. Candidates retrieval results per country

Country code	Recall (only location)	Recall (only neighbor-hood)	Recall (only category neighbor-hood)	Recall (only text similarity)	Recall (all together)	Average number of ground true duplicates	Average number of candidates	Number of points per country
MY	0.926	0.91	0.943	0.899	0.982	1.864	89.771	46,326
MX	0.951	0.915	0.962	0.895	0.987	1.882	70.584	21,396
BR	0.966	0.936	0.98	0.944	0.995	1.586	70.248	51,241
BE	0.954	0.931	0.964	0.897	0.991	1.921	72.38	25,821
CA	0.977	0.956	0.984	0.944	0.997	1.628	63.866	11,927
TH	0.919	0.903	0.925	0.848	0.983	2.08	102.401	58,808
FI	0.97	0.956	0.974	0.898	0.992	2.163	71.697	6634
ES	0.976	0.951	0.983	0.947	0.996	1.689	64.847	12,923
ID	0.751	0.73	0.757	0.824	0.912	6.62	93.234	110,796
IT	0.974	0.942	0.98	0.931	0.995	1.788	62.005	11,395
DE	0.966	0.937	0.976	0.916	0.994	1.912	65.105	17,813
SA	0.948	0.892	0.96	0.838	0.985	2.116	66.296	10,867
HK	0.925	0.916	0.932	0.861	0.981	1.821	104.116	5076
IN	0.983	0.94	0.988	0.93	0.996	1.653	59.191	4646
GR	0.974	0.935	0.98	0.907	0.991	1.918	63.777	5850
KR	0.933	0.903	0.951	0.862	0.986	1.993	80.709	19,087
SG	0.92	0.931	0.915	0.903	0.981	1.882	127.155	21,005
GB	0.954	0.931	0.96	0.912	0.992	1.777	67.804	25,544
RU	0.928	0.895	0.936	0.771	0.975	3.576	77.542	57,030
PH	0.858	0.856	0.908	0.861	0.972	2.42	103.496	22,180
FR	0.967	0.938	0.971	0.905	0.99	2.061	67.904	14,174
JP	0.955	0.947	0.959	0.921	0.993	1.861	95.703	70,032
CN	0.953	0.871	0.967	0.894	0.99	1.862	64.247	7445
TR	0.888	0.852	0.91	0.834	0.963	3.811	87.074	115,177
UA	0.96	0.924	0.975	0.86	0.992	2.034	71.425	9864
CL	0.97	0.96	0.983	0.958	0.996	1.411	76.547	7788
CZ	0.975	0.954	0.979	0.905	0.995	1.749	67.912	4920
AR	0.867	0.846	0.883	0.862	0.91	2.831	68.304	5255
US	0.955	0.926	0.961	0.928	0.992	1.866	76.128	245,284
AU	0.956	0.937	0.968	0.928	0.994	1.648	68.044	10,449
NL	0.968	0.949	0.973	0.932	0.994	1.813	64.834	7713
others	0.96	0.923	0.969	0.91	0.99	1.988	66.703	94,335

The final metrics are presented in Table 3. The final aggregated metric for our solution was 0.888, which was increased to 0.905 using post-processing logic presented in Sect. 2.4.

As a result, we achieved the top 4% result among 1079 teams and got a silver medal in the presented competition.

4 Discussion

We developed an end-to-end solution that can process a huge amount of data and de-duplicate it using a limited amount of computational resources. Usually, the solution is complex as it needs to use classical machine learning, geospatial data processing techniques, and advanced solutions in NLP.

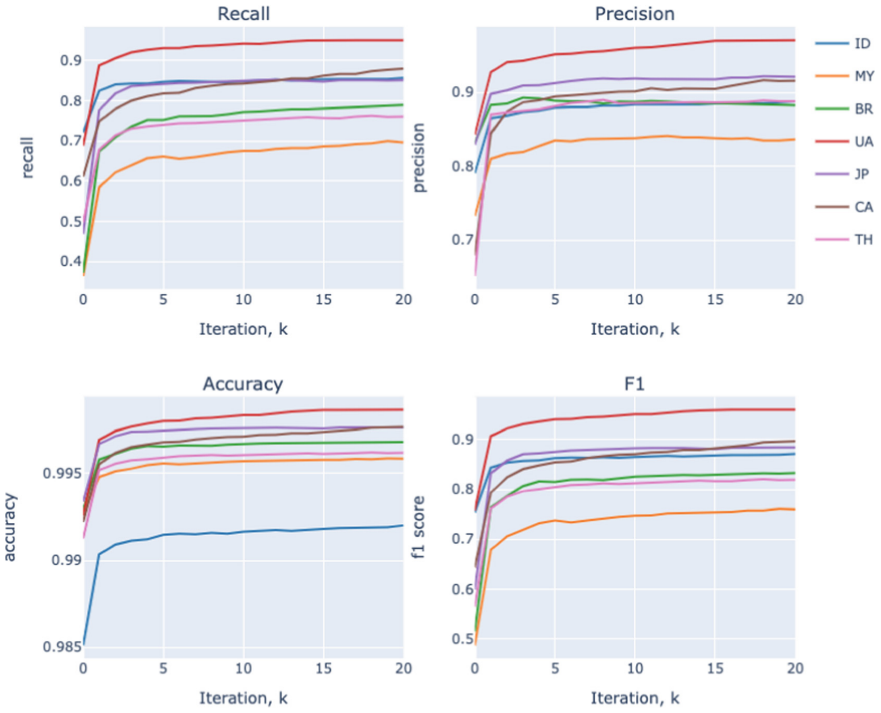

Fig. 5. Validation metrics for sampled countries

We achieved outstanding results for our specific task, but this solution has numerous limitations that we want to discuss. That can be an idea for further research in presented the field.

1. Data specifics are important. We assume that such an approach can work for similar datasets, but with severe limitations. We observed that in our case, text and category similarity had great importance in duplicate findings. However, it is important to mention that presented data is artificially generated, so in real life, the situation can be different. For further research, it is important to generate more specific geospatial features like collocations and neighborhood analysis [8]. That is also important to test the solution on other related datasets.
2. In real life, there can be a bigger amount of data, that requires processing using distributed computing. One of the lines for further research can be adapting the presented solution for it similar to previous works [1].
3. Our solution presents a multistage modeling approach. It can result in error accumulation. A possible further line of research is building a solid solution to reduce this effect.

Table 3. Validation metrics for end-to-end solution

Country code	Recall	IoU	Number of true duplicates	Number of predicted duplicates	Number of samples per country
MY	0.905	0.883	1.937	1.582	46,326
MX	0.944	0.927	1.884	1.754	21,396
BR	0.951	0.932	1.59	1.5	51,241
BE	0.952	0.935	1.955	1.795	25,821
CA	0.982	0.978	1.639	1.597	11,927
TH	0.892	0.859	2.089	1.833	58,808
FI	0.978	0.975	2.181	2.097	6634
ES	0.98	0.975	1.713	1.619	12,923
ID	0.827	0.764	6.723	3.683	110,796
IT	0.978	0.971	1.798	1.739	11,395
DE	0.973	0.963	1.93	1.848	17,813
SA	0.943	0.93	2.181	1.976	10,867
HK	0.949	0.947	1.92	1.722	5076
IN	0.992	0.989	1.677	1.636	4646
GR	0.981	0.979	1.998	1.786	5850
KR	0.936	0.919	2.039	1.841	19,087
SG	0.899	0.886	1.986	1.552	21,005
GB	0.966	0.953	1.817	1.701	25,544
RU	0.892	0.859	3.628	2.611	57,030
PH	0.905	0.875	2.442	2.12	22,180
FR	0.961	0.953	2.332	1.869	14,174
JP	0.943	0.922	1.866	1.764	70,032
CN	0.964	0.958	1.92	1.812	7445
TR	0.877	0.826	3.821	2.706	115,177
UA	0.96	0.954	2.102	1.928	9864
CL	0.982	0.98	1.414	1.37	7788
CZ	0.99	0.988	1.757	1.733	4920
AR	0.902	0.901	2.856	1.634	5255
US	0.935	0.907	1.872	1.733	169,958
AU	0.98	0.977	1.672	1.601	10,449
NL	0.98	0.976	1.914	1.761	7713
Other	0.926	0.896	2.051	1.744	94,335

References

1. Cousseau, V., Barbosa, L.: Linking place records using multi-view encoders. Neural Comput. Appl. **33**(18), 12103–12119 (2021). https://doi.org/10.1007/s00521-021-05932-9
2. Dalvi, N., et al.: Deduplicating a places database. In: Proceedings of the 23rd International Conference on World Wide Web. WWW '14, pp. 409–418. Association for Computing Machinery, Seoul, Korea (2014). ISBN: 9781450327442. https://doi.org/10.1145/2566486.2568034
3. Dolatshah, M., Hadian, A., Minaei-Bidgoli, B.: Ball*-tree: efficient spatial indexing for constrained nearest-neighbor search in metric spaces (2015). arXiv preprint arXiv:1511.00628
4. Dorogush, A.V., et al.: Fighting biases with dynamic boosting (2017). arXiv:1706.09516
5. Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs (2016). arXiv:1603.09320
6. Mikolov, T., et al.: Efficient estimation of word representations in vector space. In: Proceedings of Workshop at ICLR 2013 (2013)
7. Reimers, N., Gurevych, I.: Sentence-BERT: sentence embeddings using Siamese BERT-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (2019). arXiv:1908.10084
8. Scellato, S., Noulas, A., Mascolo, C.: Exploiting place features in link prediction on location-based social networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '11, pp. 1046–1054. Association for Computing Machinery, San Diego, California, USA (2011). ISBN: 9781450308137. https://doi.org/10.1145/2020408.2020575
9. Yang, C., et al.: Place deduplication with embeddings. In: The World Wide Web Conference. WWW '19, pp. 3420–3426. Association for Computing Machinery, San Francisco, CA, USA (2019). ISBN: 9781450366748. https://doi.org/10.1145/3308558.3313456