Neural Computing and Applications manuscript No.
(will be inserted by the editor)

Linking Place Records Using Multi-view Encoders

Vinícius Cousseau · **Luciano Barbosa** ✉

Received: date / Accepted: date

Abstract Extracting information about Web entities has become commonplace in the academy and industry alike. In particular, data about places distinguish themselves as rich sources of geolocalized information and spatial context, serving as a foundation for a series of applications. This data, however, is inherently noisy and has several issues, such as data replication. In this work, we aim to detect replicated places using a deep-learning model, named PlacERN, that relies on multi-view encoders. These encoders learn different representations from distinct information levels of a place, using intermediate mappings and non-linearities. They are then compared in order to predict whether a place pair is a duplicate or not. We then indicate how this model can be used to solve the place linkage problem in an end-to-end fashion by fitting it into a pipeline. PlacERN is evaluated on top of two distinct datasets, containing missing values and high class imbalance. The results show that: (1) PlacERN is effective in performing place deduplication, even on such challenging datasets; and (2) it outperforms previous place deduplication approaches, and competitive algorithms, namely Random Forest and LightGBM using pairwise features, on both datasets in regards to different metrics (F-score, Gini Coefficient and Area Under Precision-recall Curve).

Keywords Places · Record Linkage · Deep Learning · Representation Learning

Vinícius Cousseau
Incognia, Palo Alto, California, USA
E-mail: vinicius.cousseau@incognia.com

Luciano Barbosa ✉
Universidade Federal de Pernambuco, Recife, Pernambuco, Brazil
E-mail: luciano@cin.ufpe.br

1 Introduction

Collecting and managing data from the Web provides a way for applications to access a wide range of data domains and plentiful records. Due to many factors, it is a process which suffers from issues pertaining to a noisy and unstructured nature. While the amount of information covered by the Web grows steadily, only 60.3% of its websites display their data in some structured manner [50]. On top of that, one may also encounter well-structured but ill-defined data, which not only is harder to treat, but also lessens the informational value of it.

In this paper, we deal with data about places around the world. These places are gathered mostly from the Web by a focused Web crawler, and each of them represents a real physical space with a given name, location, and several other attributes which are further presented in this work. Thus, places are a rich source of location-sensitive information and geospatial context, with data providers such as Factual [1] and Safegraph [2] collecting them for commercial purposes.

Through this context, they spark interest in applications such as recommendation platforms[3], where users may post some reviews and receive suggestions on places to visit, and check-in apps[4], where users may check-in to a place and share their experience. Places also play a central role in the analysis of users' behavior patterns for engagement solutions, such as the one developed by Inloco[5].

One of the most critical normalization issues that must be addressed when dealing with places collected from the Web is replicated data. Replicated places in a database may lead to problems such as fragmentation of user activity in

[1] https://www.factual.com/data-set/global-places/
[2] https://www.safegraph.com/
[3] https://www.tripadvisor.com/
[4] https://www.swarmapp.com/
[5] https://www.inloco.com.br/solutions

analytical systems, where the visitation pattern of a user may be wrongly determined. For instance, if an analytical system depends on the frequency of visits to a category of place by a given user to infer their preferences, and a restaurant entity is replicated N times in its database, the system could account for N separate visits to a restaurant instead of a single one, making this user's preferences misleading.

Record linkage, also commonly known as entity resolution or entity matching, is a field which refers to the detection of redundant data [19]. This field has evolved considerably since its inception, as the problem not only persists but is also enhanced, because the Web is mostly open to user input and there are usually a plethora of websites which display the same kind of information in different forms. Record linkage is commonly divided into three major steps: blocking, where records are aggregated based on some common characteristic to improve scalability; pairwise matching, in which pairs of records are matched against each other to detect duplications; and clustering, where pairs deemed as duplicates in the pairwise matching step are grouped. Among those, the pairwise matching of entities plays a central role in typical record linkage pipelines [3,2,11].

The detection of replicated place records is affected by toponym ambiguity [6], that is, different names for the same real-world place, such as synonyms, abbreviations, and representations of a toponym in a different language (transliterations), or different places with the same name in different geographical positions. This ambiguity adds to the noise present in Web data to make textual information of place entities cumbersome to deal with. Another challenge to perform this task is that duplicate cases are uncommon, and thus handling class imbalance gracefully is important for any solution to this problem.

Using a single data field to perform pairwise matching, such as addresses or phone numbers, is not reliable as their coverage on the Web may be low, and it may also lead to biased matches producing imprecise duplicate detection. For instance, even though two places with the same phone number may be duplicates in a dataset, they may also be places from a same chain store sharing a customer service central phone number. Hence, representing places as a composition of fields is beneficial to build a more robust pairwise matching solution.

A popular approach for record linkage and similar matching tasks in the recent years is utilizing machine learning techniques to perform comparisons between records or representations of them [25,23,53,45,44,32,2,56]. Yang *et al.* [56], for instance, attempt to first build generic representations for places in an unsupervised manner and use them for the deduplication task in a supervised manner. As opposed to them, we devise a deep neural network named PlacERN (Place Entity Resolution Network) that builds representations in a supervised manner for pairs of places, directly on the deduplication task. More specifically, this network learns distinct representation encoders based on fields of a place: its name, address, geographical coordinates, and categories. These representations are combined in an affinity layer by concatenating their vectors and calculating different types of similarities between them. Finally, a fully-connected network with a final sigmoid layer receives the affinity layer's output and predicts the likelihood of a pair of places being a duplicate, serving as the core driver behind a full-blown record linkage pipeline.

We consider the contributions of this paper to be as follows:

- As part of a working system [10], we develop PlacERN, a deep neural network which is able to capture information from multiple place fields and use it to generate a representation for a place pair. This representation is then utilized to classify a given pair into duplicate or non-duplicate, and handles missing values by design;
- We train and evaluate PlacERN against both PRF and PLGBM, two supervised models developed in this work on a set of pairwise features, and other methods in the state-of-the-art, with industrial datasets having instances with missing values and imbalance ratios from 8.24 to 24.0. In all scenarios, PlacERN is shown to surpass all other compared models in terms of $F_{\beta=0.5}$, normalized Gini coefficient, and Area Under Precision-recall Curve. PRF and PLGBM are also shown to outperform other baseline methods;
- We perform feature importance studies using a game theoretic approach (Shapley Additive Analysis [31]) on top of PRF and PLGBM in order to drive the development of PlacERN, showing that most gains are to be expected from places' names and addresses;

2 Background

This section serves as a foundation for understanding the data records dealt with by our proposed solution. It provides explanations on what a place entity is in the context of this work, which attributes are commonly associated with it, and issues pertaining to each attribute. We use these concepts to lay out our research problem.

Place Entity Definition Looking at previous works, we see that the "place" term is used ambiguously when referring to an entity type in a database. For instance, some works define places as any entity labelled as such in a specific Web service [13,56], while others simply as Points of Interest (POI) [55, 17].

These inconsistencies derive from the intrinsic characteristics of each database and from the business interests involved in common place services. While we do not aim to

tackle these differences in nomenclature, the definition of a place is the context of our work is given by:

Definition 1 [Place entity]. A place entity in our database is a direct representation of a physical location in the real world which has well-defined boundaries, a clear purpose for visitation, and an area sufficient for a person to be enclosed in.

Some examples of places that fit this definition are stores, shopping malls, restaurants, and parks. Meanwhile, stretches of sands on a beach, statues, and building floors are not.

2.1 **Place Fields**

The place fields utilized in this work, which are also encountered in other Web-based place datasets, are an *id* (a unique textual identifier), a name n, a geo location g (expressed in latitude and longitude coordinates), a textual address r, a list of categories c, a *phone_number*, a *homepage* URL, and a *parent_place_id* (which is an *id* of another place which encompasses the current one, if any). Only the *id*, the name n, and the location g are obligatory in our work, and we overload the address nomenclature to refer to thoroughfares (street names) and sub thoroughfares (complements) only.

This collection of fields presents several issues which may be encountered in any Web-based database in this domain. Textual fields, for instance, suffer from normalization problems like toponym ambiguity. The geo location field presents issues due to a common lack of precision resulting from the geocoding - address to latitude and longitude translation - strategies utilized by websites, or noise introduced by GPS sensors in indoor environments - utilized by users to create new places in the Web. Finally, the *parent_place_id* and categories are generated in our work by a heuristic utilizing name patterns and a multi-label classification model, respectively. As such, they suffer from errors stemming from these generational strategies.

2.2 **Problem Definition**

We then define our research problem as follows:

Definition 2 [Research Problem]. Given two place entities P_a and P_b, each of them having a name n and a geographical location g, with possible inclusions of an address r, a list of categories c, a parentage id *parent_place_id*, a *homepage*, and a *phone_number*, we aim to detect whether they represent the same real-world place ($P_a \sim P_b$).

3 PlacERN

In this section, we introduce PlacERN (Place Entity Resolution Network), a deep neural network that performs pairwise matching to detect replicated places. For that, PlacERN captures multi-level information for each place using intermediate mappings and non-linearities, and then compares these representations in order to predict whether a place pair is a duplicate or not.

The deep neural network architecture is laid out in Figure 1. For each place pair, the network receives its names n, addresses r, category lists c, and geographical information g in the form of latitude and longitude pairs. Then, to capture different views of the input, each place is processed by four different encoders that transform its original representation into ones more suitable for the classification task. More specifically, word and character encoders are applied on each place's textual fields (name and address), a category encoder on the category list, and a geographical encoder on the Haversine distance calculated from the places' latitude and longitude.

Next, the different representations produced by the encoders for each place, with the exception of the geographical one, are concatenated and compared against each other in an Affinity Generation step in order to generate similarity values using different metric strategies. The resulting merged tensor is then concatenated with the geographical distance embedding generated by the geographical encoder, and passed through a feed-forward network with dropout regularization between each pair of layers. Lastly, the final tensor is processed by a sigmoid-activated layer with a single neuron, generating a probabilistic output which represents the likelihood of the input pair of places being duplicates.

In the remainder of this section, we explain each encoder of the proposed architecture in more detail, alongside the final Affinity Generation steps. We also shed some light on failed attempts to improve our model, so as to achieve a better explanation of our thought process and assist other researchers.

3.1 Word Encoder

The word-level representation encoder attempts to capture features of words in each place's name n^{P_i} and address r^{P_i}, with $i \in \{a, b\}$, producing a word-level place representation $e_w^{P_i}$. By learning representations of these word sequences for place pairs marked as duplicates, it is expected that words that appear in similar contexts and represent duplications reside closely in the embedded space, thus pulling both places in a pair closer.

To do that, it first transforms the name and address of places P_a and P_b into indexed padded sequences of size S^w.

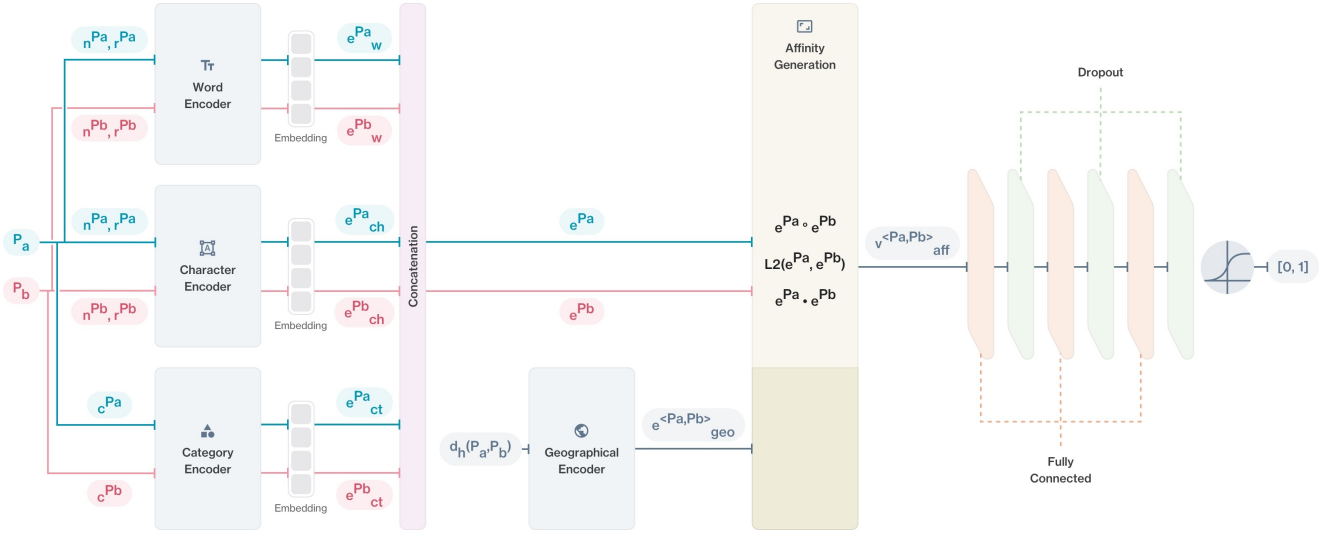


Fig. 1: Diagram of our deep neural network architecture for classifying place pairs, PlacERN. The network receives names, addresses, category lists, and geographical information from place pairs as input. Four distinct encoders and an affinity generation step process this data to build a representation for the pair. This representation is passed through a feedforward network with a final sigmoid layer, producing a probabilistic result.

They are then fed to separate regular word embedding layers for names and addresses. The word embeddings of these sequences are initialized with pre-trained vectors of m dimensions to take advantage of transfer learning:

- For **name embeddings**, a simple case-insensitive skip-gram model [34] was trained on top of the *corpus* of all tokenized place names in our dataset;
- For **address embeddings**, we utilized case-sensitive embeddings trained with the FastText model [4] on top of a Wikipedia and Common Crawl *corpus* [22].

The reasoning behind the different embedding sources for names and addresses is that place names may contain unique words, not seen much in other *corpora* because they are relevant in establishing an identity for the place. Meanwhile, the presence of these unique words in addresses is unusual. As a result of that, training an address embedding on top of our dataset does not yield better results while imposing an additional computation step.

After producing sequences of embeddings for names α^{P_i} and addresses β^{P_i} - with every embedding vector $\{\alpha,\beta\}_j^{P_i} \in \mathbb{R}^m$ - for places P_a and P_b respectively, the encoder passes them through a siamese bidirectional GRU [7] layer. Thus, our network contains two GRUs: GRU_a and GRU_b, with a shared number of dimensions l, weights, and parameters. Embedding sequences for place P_i are passed through GRU_i both in order and in reverse order, and the concatenated hidden states from the last GRU cells for each ordering are used as output.

Hence, $GRU_i(\alpha^{P_i})$ produces name word embeddings $e_{n_w}^{P_i} \in \mathbb{R}^{2l}$ and $GRU_i(\beta^{P_i})$ produces address word embeddings $e_{r_w}^{P_i} \in \mathbb{R}^{2l}$. Finally, $e_{n_w}^{P_i}$ and $e_{r_w}^{P_i}$ are concatenated to form the place word level embedding $e_w^{P_i} \in \mathbb{R}^{4l}$.

3.2 Character Encoder

The character-level encoder aims to build a character-based representation $e_{ch}^{P_a}$ for P_a and $e_{ch}^{P_b}$ for P_b for a given pair: P_a and P_b. Usage of this type of encoder adds resilience to normalization errors and edge cases encountered with words, such as *International* and *Intl.*; *HulaHoop* and *Hula Hoop*.

Given places P_a and P_b, their names n^{P_i} and addresses r^{P_i} are first transformed into indexed padded character sequences of size S_n^{ch} for names and S_r^{ch} for addresses. Akin to the word encoder, we use two separate embedding layers for name and address characters, producing embeddings of m dimensions. Both layers are initialized with random weights, which lets the character encoder receive each place pair's indexed sequences as input.

Afterwards, the sequence of name character embeddings Φ^{P_i} and address character embeddings Ψ^{P_i} - with every embedding vector $\{\phi,\psi\}_j^{P_i} \in \mathbb{R}^m$ - for places P_a and P_b respectively, are each passed through distinct 1D-CNNs [9], with the same kernel size k and filters $F \in \mathbb{R}^{k\times d}$, but different random weight initialization. For the sake of simplicity, we name the 1D-CNN which processes name characters as CNN_n and the 1D-CNN which processes address characters as CNN_r.

When used with textual data, the 1-dimensional convolution operations performed by 1D-CNNs may be interpreted as window-based feature extractors. In addition, they have

shown to be efficient in dealing with sequences of noisy data [28,51], and present better time efficiency as character feature extractors when compared to RNNs due to the larger sequence length of characters.

Consequently, we have that $CNN_n(\Phi^{P_i}) = E^{P_i}_{n_{ch}}$, with $E^{P_i}_{n_{ch}} \in \mathbb{R}^{(S^{ch}_n - k+1)\times d}$, and that $CNN_r(\Psi^{P_i}) = E^{P_i}_{r_{ch}}$, where $E^{P_i}_{r_{ch}} \in \mathbb{R}^{(S^{ch}_r - k+1)\times d}$. These internal representations from the 1D-CNNs are then passed through a global max pooling layer which generates the name character embeddings $e^{P_i}_{n_{ch}}$ and address character embeddings $e^{P_i}_{r_{ch}}$ for each place, with $e^{P_i}_{\{n_{ch},r_{ch}\}} \in \mathbb{R}^d$. In its final step, this layer performs concatenations $e^{P_i}_{n_{ch}} \circ e^{P_i}_{r_{ch}}, i \in \{a,b\}$ to generate the final character embeddings $e^{P_a}_{ch}$ and $e^{P_b}_{ch}$, with $e^{P_i}_{ch} \in \mathbb{R}^{2d}$.

3.3 Category Encoder

Our category encoder deals directly with the category list c of each place $P_i, i \in \{a,b\}$. Its goal is generating a category level embedding $e^{P_i}_{ct}$, following the intuition that places belonging to the same or similar categories have a higher change of being duplicates than those belonging to dissimilar categories.

Even though places may present their category in their own names or addresses, e.g. *John's Bar*, both the word and character encoders can be incapable of capturing the necessary latent patterns required in order to discern between places of different categories. Moreover, some places such as a sunglass chain store named *Chilli Beans* may be misleading. Category information, however, is not a standalone way of detecting duplicate places, and should be seen only as a way to reinforce certain patterns.

The categories of input places P_a and P_b, which pertain to a list of 122 pre-defined values, are first transformed into index sets of a fixed size S^{ct}. These sets are then consumed by an embedding layer initialized with m-dimensional vectors for each category, producing the embedding sets Θ^{P_a} and Θ^{P_b}. After being generated, the embedding sequences Θ^{P_a} and Θ^{P_b} undergo a global max pooling operation, followed by an L2 normalization. This pipeline results in embeddings $e^{P_a}_{ct} \in \mathbb{R}^m$ and $e^{P_b}_{ct} \in \mathbb{R}^m$.

The embedding layer is initialized before training begins by using the pre-trained word embeddings for place names mentioned in Section 3.1, as it provides better initial results than using random initialization. More specifically, for each word in a category name, their embeddings are summed and L2-normalized to create an initial embedding space in which categories with contextually similar words are close. For instance, given the category *hardware_store*, we construct an initial embedding $\theta_{hardware\_store} = L2(pre_{hardware} + pre_{store}), \theta_{hardware\_store} \in \mathbb{R}^m$, where *pre* are the pre-trained word embeddings. We highlight that the aforementioned step is performed *a priori*, and does not depend on our network being trained beforehand.

3.4 Geographical Encoder

The last encoder in our architecture is the geographical one, whose goal is generating an embedding $e^{<P_a,P_b>}_{geo}$ for each pair of places, which encapsulates the properties pertaining to the geographical distance between them. It operates on top of the Haversine distance d_h between the two places, which is already a pairwise metric. Duplicate places usually do not have the same latitude and longitude information due to the errors described in Section 2, and places closer to each other should usually have a higher chance of being duplicates than places which are far apart.

As Xiang *et al.* [26] show, discretizing real values with a good subdivision strategy is sufficient for a network to capture the latent patterns of said values without losing information. So as to discretize the distances between places, we firstly calculate the maximum distance amongst all place pairs in the dataset. By trivially assuming a minimum distance of 0 meters, we then create B distance buckets by equal-width discretization. Each bucket represents an interval of distances $[b_t, b_{t+1}[$, and an embedding layer is randomly initialized for each of those buckets with m dimensions. Finally, for a pair of places P_a and P_b, their distance is calculated by $d_h(P_a,P_b)$, they are attributed to the respective bucket, and then the geographical encoder outputs a final embedding $e^{<P_a,P_b>}_{geo} \in \mathbb{R}^m$.

3.5 Affinity Generation

With all of the embeddings $e^{\{P_a,P_b\}}_w$, $e^{\{P_a,P_b\}}_{ch}$, $e^{\{P_a,P_b\}}_{ct}$, and $e^{<P_a,P_b>}_{geo}$ produced by the encoders for a pair of places P_a and P_b, PlacERN then proceeds to perform comparisons in order to learn a similarity metric between the two places. Before comparing the embeddings, however, the network performs a merging operation on the embeddings pertaining to a single place - namely every one but the geographical - to produce an appropriate embedding for each place. Hence, we form the place embeddings $e^{P_i} = e^{P_i}_w \circ e^{P_i}_{ch} \circ e^{P_i}_{ct}$, with $e^{P_i} \in \mathbb{R}^{4l+2d+m}$ and $i \in \{a,b\}$.

Next, comparisons between the place embeddings are performed by a series of operations in a merge layer:

- A concatenation of the two embeddings, $e^{P_a} \circ e^{P_b}$;
- An element-wise L2 distance between the two embeddings, $(e^{P_a} - e^{P_b})^2$;
- An element-wise multiplication of the two embeddings, $e^{P_a} \cdot e^{P_b}$.

These results are thus concatenated into a tensor $v^{<P_a,P_b>}_{me} = e^{P_a} \circ e^{P_b} \circ (e^{P_a} - e^{P_b})^2 \circ (e^{P_a} \cdot e^{P_b})$, with the tensor $v^{<P_a,P_b>}_{me}$

$\in R^{16l+8d+4m}$. Subsequently, the geographical distance embedding $e_{geo}^{<P_a,P_b>}$ is concatenated with the merge result since it already represents a merge operation itself, resulting in the final merged tensor $v_{aff}^{<P_a,P_b>} = v_{me}^{<P_a,P_b>} \circ e_{geo}^{<P_a,P_b>}$, with $v_{aff}^{<P_a,P_b>} \in \mathbb{R}^{16l+8d+5m}$.

Afterwards, the $v_{aff}^{<P_a,P_b>}$ tensor is passed through a feed-forward network of L fully connected layers using the ReLU activation function [36]. The first of these layers has H_0 neurons, and each of the following j ones has $H_j = \frac{H_0}{2^j}$ neurons, where $j \in [1,L]$. For ease of explanation, we name the results from each of the dense layers as $v_j^{<P_a,P_b>}$. To mitigate overfitting, we also add Dropout regularization [47] with a fixed ratio between each pair of fully connected layers in the feed forward network.

Finally, the resulting tensor from the last dense layer in the feed forward network, $v_L^{<P_a,P_b>}$, is sent to a single neuron layer with a sigmoid activation function. It produces a probability which represents the likelihood of $P_a \sim P_b$. A threshold t_c is then applied to generate a binary classification result. Summarizing, the final output of PlacERN is defined by the following equation:

$$o^{<P_a,P_B>} = \begin{cases} 1 & \text{if } \sigma(W_l \cdot v_L^{<P_a,P_b>} + b) \geq t_c \\ 0 & \text{if } \sigma(W_l \cdot v_L^{<P_a,P_b>} + b) < t_c \end{cases} \quad (1)$$

where $W_l \in \mathbb{R}^{1 \times \frac{H_0}{2^L}}$ is the final layer's weight matrix, b is the layer's bias, and t_c is the decision threshold. The decision threshold t_c may be tuned according to some quality metric. A final output value of 1 means $P_a \sim P_b$, and a value of 0, otherwise.

3.6 Model Training

To train the model while accounting for the expected high imbalance factor in the record linkage task, the network minimizes the Focal Loss training function [30]. This function expands the traditional binary cross-entropy one by adding a modulating factor with a tunable focusing parameter γ. It leads the model to focus on samples which are presenting more error than the others, and reduces the contribution brought by easier, more abundant samples. Mathematically, the focal loss of a sample labeled as $y \in \{0,1\}$, with prediction p, is defined by:

$$\text{FL}(p,y) = \begin{cases} -(1-p)^\gamma \log(p) & \text{if } y = 1 \\ -p^\gamma \log(1-p) & \text{if } y = 0 \end{cases} \quad (2)$$

A class-balanced focal loss based on the effective number of samples [12] was also analyzed as an option, but failed to improve results during initial experiments.

3.7 Failed Attempts in PlacERN

Throughout the development of PlacERN, we attempted several approaches to build and improve our results that were met with failure. Thus, this section highlights the most impactful of said failures, in order to help researchers facing similar issues.

Word Encoder The first failed attempt to highlight is the usage of publicly available FastText embeddings [33] for place names. Due to the unique nature of place names, the pre-trained embeddings and the out-of-vocabulary generated tokens were shadowed by the skip-gram model trained on our own *corpus*.

Next, we attempted different architectures of siamese GRU networks for the word encoder: separate GRUs for names and addresses, max pooling over GRU hidden states, and stacked GRUs with skip connections [44]. The increase in model complexity brought by these approaches, however, did not proportionally increase our representational power, which was a trigger for increasing variance.

Another attempt to improve our word level encoder was using co-attention from Xiong *et al.* [53], following the intuition that it could be a natural extension of the concept of core words for places [13]. However, this implementation lowered our performance, while slowing down the process of training the network and adding more complexity to the solution.

Character Encoder Siamese GRU networks were the first attempt at implementing the character encoder. The greater length of character sequences, however, imposed a bottleneck in the network, increasing training times tenfold, and did not improve the evaluation metrics. Thus, 1D-CNNs are used as an alternative, as described in Section 3.2.

Geographical Encoder A possible alternative for inserting geographical context into a deep network is transporting latitude and longitude pairs to an embedded space which captures the characteristics of their surroundings. We implemented this approach by grid subdivision and building embeddings for each grid, but the region embeddings did not translate to the duplicate detection case. Some grids also had insufficient data. We also attempted a sequence encoding approach [51], but the original work uses sequences with lengths upwards of 50, while ours had a fixed size of 2, and thus it was unable to provide improvements.

4 Record Linkage Pipeline

To utilize PlacERN in our working system [10], the first step is generating place pairs from a given dataset. Generating all possible pairs, however, incurs in a complexity

of $O(n^2)$, where n is the size of the places dataset. Thus, a record blocking [48] step should be performed to reduce the number of pairs whilst preventing the eager elimination of true duplicate pairs. In this work, we utilize a MapReduce [15] blocking step consisting of:

1. A partitioning phase, where the places set is partitioned into Geohashes [35,38] with a precision of 6 characters, and all possible pairs are generated inside each Geohash only;
2. A filtering phase, where a Jaro-Winkler similarity metric [52] is computed for each pair and a similarity threshold λ is applied to filter trivial non-duplicates.

The usage of Geohashes as partitioning keys translates naturally to the MapReduce computation model, and thus we are able to use distributed computation by means of Apache Spark [57]. With the record blocking step generating the potential duplicates, PlacERN is then able to be executed for the purposes of this work. However, the replication factors encountered in Web-based datasets supersede the pairwise approach taken, i.e. replicated places may appear more than two times, and even hundreds of times [13]. To account for this, our pipeline employs a simple graph-based approach [11] to detect connected components among the pairwise results from PlacERN and extend the results to replication tuples of arbitrary sizes.

This proposed pipeline has been used and evolved in a production environment, processing tens of millions of records in a recurrent basis. Moreover, the same models utilized in the pipeline, such as PlacERN, are able to be ported to an API for solving the linkage problem in real-time environments [10].

5 Data Overview

The dataset utilized in this work is a snapshot, created *circa* 2018 both by scraping structured data, such as *schema.org*[6], from the Web and by manual insertions from an internal company team. This dataset, referred to as $Places_{DB}$ from here on, contains 28,467,486 place records with multiple languages and characteristics.

Roughly 43.28% of all places in $Places_{DB}$ are located in the US and 16.82% in Brazil, and the primary languages in those countries (English and Portuguese, respectively) present different linguistic features. Because of that, we further partition $Places_{DB}$ into a dataset of places from the US, $Places_{US}$, and a dataset of places from Brazil, $Places_{BR}$.

The attribute coverage for these datasets is 100% for the name and geo location fields, since they are obligatory. For $Places_{US}$, 97.50% of places contain an address, 70.01% contain categories, 7.33% contain a homepage, and just 0.55% contain a parent place. Meanwhile, for $Places_{BR}$, 99.89% of places contain an address, 67.83% contain categories, 2.60% contain a homepage, and just 1.07% contain a parent place. We are then able to conduct experiments taking missing data into consideration. The categorical and geographical distributions of both datasets are analyzed to assert that they are not biased toward specific categories or locations.

[6] https://schema.org/

5.1 Generating Pairs using Blocking

From these sets, we first generate pairs of places to evaluate PlacERN. To avoid generating a dataset with a quadratic size in regards to the number of records in $Places_{US}$ and $Places_{BR}$, we utilize the same blocking algorithm described in Section 4 on top of $Places_{US}$ and $Places_{BR}$. To select the λ Jaro Winkler similarity threshold used in the filtering phase of the blocking algorithm, we divide the place pairs from both place datasets into similarity buckets of range 0.1 from 0.4 to 1.0 and manually inspect 100 random pairs from each of them. Our goal was to find a threshold low enough to avoid excluding too many pairs indicating a duplication, and high enough to clean up obvious non-duplicates. Thus, these inspected pairs serve as representatives in a manual analysis only. By only finding duplicates in samples from buckets whose similarity is higher than 0.8, we set $\lambda = 0.8$ to avoid hurting recall.

5.2 Labeling

We generate labels indicating duplication or non-duplication for each of the place pairs produced by the blocking algorithm. Albeit usually generating a higher quality of labels, manually labelling all of the generated samples is infeasible given our available resources. Crowdsourced approaches through online platforms, on the other hand, present impediments in the form of a lack of support for the Portuguese language and higher error rates than a structured curatorship process [54].

In light of that, we first utilize the *phone_number* attribute, present in 27.04% of places in $Places_{BR}$ and 91.01% of places in $Places_{US}$, to generate labels for each pair, excluding it from our model to avoid leakage. This follows findings that a reasonable number of the place pairs which share a phone number represent duplications in our datasets. Hence, place pairs with phone numbers are selected from each set, have their special digits such as "+" and "-" removed, and then are compared through exact match to produce a positive or negative label.

These initial labels generated through phone match, however, are not completely precise: two places in the dataset may have different phone numbers and still be duplicates, due to a lack of constant updates to their source web pages,

and two places may also have the same phone number and not be duplicates. We notice that this second case tends to happen when places belong to a chain store, which, in many cases, have a fixed number for servicing customers in all stores. These issues, alongside the low attribute coverage in the Brazilian dataset and the Web in general, are impediments to using exclusively phone numbers to match places.

Thus, to improve the label quality, we manually curate several samples of these results, fixing erratic patterns such as different places inside a mall or belonging to the same chain store, among others. This laborious process generates the $Pairs_{US}$ and $Pairs_{BR}$ silver-truth sets, whose characteristics are displayed in Table 1.

Table 1: Characteristics of our $Pairs_{US}$ and $Pairs_{BR}$ silver truth sets.

Data set	#pairs	#places	#positive pairs	#negative pairs
$Pairs_{US}$	3,009,428	2,267,885	325,809	2,683,619
$Pairs_{BR}$	597,452	365,092	24,892	572,560

Both sets preserve a low ratio of duplicates to non duplicates. Also, we note that the $Pairs_{BR}$ set has almost three times the skew (24) of the $Pairs_{US}$ one (8.24). Since the exact ratio of duplicates in a database varies greatly over locations and over time, this difference in skew allows us to train and evaluate models under different data quality scenarios. Moreover, the general differences in both sets make our evaluation broader.

To expand our experiments, we searched for additional ground truth sets, both pairwise and containing places only. However, all datasets we found posed some kind of issue which made it impractical or not possible to use. For instance, the labelled datasets from Dalvi *et al.* [13] and Yang *et al.* [56] are both proprietary. Readers are encouraged to refer to Cousseau (2020) [10] for further details on this process.

6 Experiments

So as to assess the quality of PlacERN, we compare it with preliminary approaches and baseline methods. Both of our silver-truth sets are split into training, validation, and test sets, using a fixed split of 70%, 20%, and 10%, respectively, preserving the class distribution in each of them. Following recommendations from Ng (2019) [37], we also sample some entries from the validation set and use them as our Eyeball validation set, which allows us to probe for error patterns in our models' results so as to look for data insights.

6.1 Runtime Environment

The experiments for deep neural networks are executed in machine instances provided by Google Colab[7], containing 13Gb RAM, an Intel Xeon CPU, and a mix of NVidia K80, T4, P4, and P100 GPUs. Meanwhile, the CPU-bound tests make use of an instance with 16Gb RAM, and an Intel Core i7-7500U CPU. In addition to that, any steps requiring distributed computation by means of Apache Spark [57], such as the record blocking one, utilize 7 AWS m4.xlarge instances, each having 16Gb of memory and 4vCPUs.

6.2 Performance Metrics

Since the problem of detecting duplicate places suffers from class imbalance, generally and in the $Pairs_{BR}$ and $Pairs_{US}$ datasets, the chosen metrics need to account for that. Ergo, each model is evaluated on the task of detecting place duplications according to three main metrics.

Gini Coefficient The first of these metrics is the normalized Gini coefficient [20, 18, 14]. It attempts to measure how far apart are the probabilistic results from random guessing, operating thus directly on the probabilistic results from the models. Its normalization is performed by dividing the resulting Gini coefficient by the Gini coefficient of the ground truth set, returning then a result between 0 and 1 (best case).

F-score The second is the $F_{\beta=0.5}$ score. The β coefficient for the F-score is set as 0.5 to account for the fact that precision outweighs recall in our use case, since place pairs incorrectly classified as duplicates may lead to possibly unrecoverable states in a database.

Precision-recall Curves Finally, to provide insights on each model's capability of being tuned according to either precision or recall, we also present Precision-Recall curves and the area under them (AUC) [43]. They show the relationship between the precision and recall metrics for different classification thresholds.

We also provide the time consumed by each model during training and execution. These times highlight the capability of each model to be fitted into an end-to-end production pipeline or real-time service.

6.3 Baseline Methods

In order to evaluate PlacERN, we compare it against both previous approaches that deal with the same problem and our own supervised learning baselines trained on top of an engineered set of pairwise features.

[7] https://colab.research.google.com

Previous Approaches In these experiments, we utilize two baselines [13,56]. According to our literature review, these are the most prolific works dealing directly with deduplication of places Web data.

- Dalvi *et al.* (2014) [13]: this approach makes use of a Expectation-maximization algorithm to build probability distributions for words of a place's name, splitting them into core and background words. The core words are then used to compare if two places are duplicates. The probability of a place pair representing duplicate places is calculated directly by the core word comparison probability derived in their work. The authors also propose a dynamic programming algorithm to take edit operations into account during comparisons, but the provided pseudocode seems to be non-functional: it seems to cause an infinite recursion;
- PE - Yang *et al.* (2019) [56]: this work proposes an unsupervised learning step to generate place embeddings. Then, an MLP is trained on top of the place embeddings, and is gradually improved with: a novel contrastive loss function (PE), batch-wise hard-sampling (PEH), source-based attentive training (PEHA), and a label denoising technique (PEHAD). Our implementation adapts the PE model, outputting the euclidean distance between places in the metric space, whose complement is then interpreted as the positive class probability. We were unable to implement the full PEHAD model, partially due to either missing details or incompatibility with our datasets. Furthermore, we contacted the authors regarding the model parameters, who recommended a search on a given range of values for the negative sampling ratio in the smoothing step, among others.

None of these works, however, provide out-of-the-box implementations for their methods, so we implement our own versions. For [56], all steps are implemented with Keras 2.3.1 and Tensorflow 1.15.2, with the aid of gensim [42] for the unsupervised steps. In appendix (§ 9.1), we detail the process of choosing the hyperparameters of both solutions.

Supervised Models As a preliminary attempt to tackle the linkage problem for place records, we engineer a set of pairwise features and train supervised models on top of them. These features are:

- A Soft TF-IDF [8] between place names (name_soft_tfidf);
- A Soft TF-IDF between addresses (address_soft_tfidf);
- The Haversine distance d_h between the pair of places;
- The difference between numeric address sub thoroughfares (sub_thoroughfare_diff);
- A Jaccard similarity between categories (categories_jaccard);
- A Jaro-Winkler [52] similarity between the core word of each place's name (core_word_jaro). Our concept of core word for each place name is the one with the maximum inverse document frequency among all places, that also adheres to low frequency thresholds in the place's Geohash, i.e. a word which is relevant both globally and locally;
- Three one-hot categorical features $\in \mathbb{R}^4$ indicating the full absence, partial absence, non-match, or match between homepages (homepage_matches), parents (siblings_match), and id with parents (parent_place_match).

Each pair of places is then represented as a row vector $v \in \mathbb{R}^{18}$, which is utilized to train a Random Forest model (PRF) [5] and a LGBM model [27] (PLGBM). Each model returns a probability of a pair being a duplicate $s \in [0,1]$, and we experimentally define a classification threshold t_c to dictate which results $o^{<P_a,P_b>}$ are classified as duplicates $P_a \sim P_b$, with $s \geq t_c$, and which are classified as non-duplicates, with $s < t_c$. Further details regarding the selection of t_c and the hyperparameters of PRF and PLGBM can be found in appendix (§ 9.2).

The PRF model is implemented using the scikit-learn and pandas libraries [40,49], while PLGBM uses the LightGBM package [27]. Meanwhile, the feature generation step is implemented in Scala with Apache Spark for distributed computation.

6.4 Setup for PlacERN

PlacERN is implemented with Keras 2.3.1 and Tensorflow 1.15.2, with gensim for the pre-trained name embeddings. We utilize the ReLu activation function [36] in the feed-forward network layers. Glorot uniform initialization [21] is utilized for the character and geographical encoder embedding layers, while the embedding layers of the word and category encoders are initialized with pre-trained embeddings, as mentioned in Section 3. For the CNN layers in the character encoder and the RNN layers in the word encoder, the hyperbolic tangent is utilized as an activation function. Training is performed in batches with the ADAM optimizer [29]. Details on its hyperparameters are found in appendix (§ 9.3).

6.5 Hyperparameter Tuning

Hyperparameters are tuned with the validation set in regards to the normalized Gini coefficient first, mostly through a Bayesian optimization process. Then, the probabilistic outputs are transformed into binary labels by applying a threshold t_c, chosen by optimization of the $F_{\beta=0.5}$ score in a separate grid search. The full optimization process and the hy-

perparameters for each model and each dataset are described at depth in appendix (§ 9).

6.6 Results

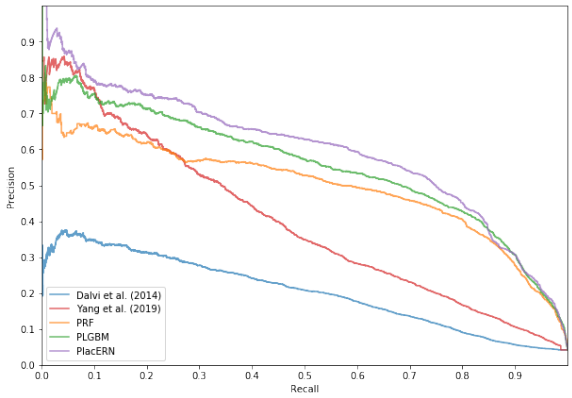
The results comparing each of our models with the baseline methods in the $Pairs_{BR}$ and $Pairs_{US}$ test datasets are displayed in Table 2. Figure 2 expands on that by displaying plots of the precision-recall curves for all applicable models on both datasets.

As presented in Table 2, PlacERN obtains the best values of $F_{\beta=0.5}$, normalized Gini score, and *AUC* in both datasets. More specifically, its $F_{\beta=0.5}$ for $Pairs_{BR}$ and $Pairs_{US}$ are 0.609 and 0.809, respectively, while its normalized Gini coefficients are 0.929 and 0.959. By breaking up the $F_{\beta=0.5}$ score into its precision and recall components, the results show that PlacERN also reaches the best precision in $Pairs_{BR}$ (0.617) and the best recall in $Pairs_{US}$ (0.712). By comparing PlacERN with PLGBM, which has the second-best results in both datasets in terms of $F_{\beta=0.5}$, we see relative gains upwards of 7.14% in the $Pairs_{BR}$ dataset and 5.61% in the $Pairs_{US}$ dataset.

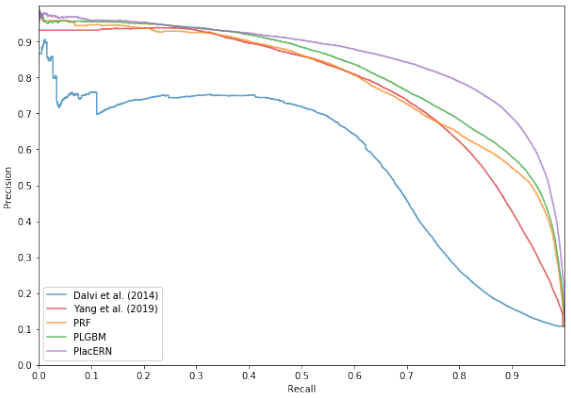
In $Pairs_{US}$, PlacERN has an AUC of 0.857 against 0.816 from PLGBM and 0.775 from PE. Meanwhile, for $Places_{BR}$, PlacERN has an AUC of 0.606 against 0.552 from PLGBM and 0.396 from PE. Analyzing the AUC alongside Figures 2a and 2b further shows that PlacERN offers an improvement over its competitors in almost all precision-recall ratios, that is: it is able to be more thoroughly tailored to attend specific demands of precision or recall ratios. For instance, fixing precision at 0.8, PLGBM has a recall close to 0.7 while PlacERN reaches a value close to 0.8 in the $Pairs_{US}$ dataset. These results confirm that PlacERN is able to better capture the place features with its encoders, independently of the size of the training dataset or its imbalance factor.

Table 2: Comparison of all models on test samples from $Pairs_{BR}$ and $Pairs_{US}$.

Dataset	Model	Precision	Recall	$F_{\beta=0.5}$	Gini	AUC
$Pairs_{BR}$	Dalvi *et al.* (2014) [13]	0.300	0.240	0.285	0.601	0.204
	PE - Yang *et al.* (2019) [56]	0.590	0.246	0.462	0.778	0.396
	PRF	0.498	**0.581**	0.513	0.915	0.498
	PLGBM	0.616	0.412	0.560	0.924	0.552
	PlacERN	**0.617**	0.542	**0.600**	**0.929**	**0.606**
$Pairs_{US}$	Dalvi *et al.* (2014) [13]	0.750	0.240	0.526	0.693	0.568
	PE - *Yang et al.* (2019) [56]	0.837	0.547	0.757	0.896	0.775
	PRF	**0.844**	0.543	0.760	0.930	0.795
	PLGBM	0.801	0.652	0.766	0.938	0.816
	PlacERN	0.837	**0.712**	**0.809**	**0.959**	**0.857**



(a) Precision-recall curves in $Pairs_{BR}$.



(b) Precision-recall curves in $Pairs_{US}$.

Fig. 2: Overlapping precision-recall curves in $Pairs_{BR}$ and $Pairs_{US}$ for all possible models.

However, by analyzing the precision-recall curves, we see that PLGBM has equivalent metrics to PlacERN at some points. Furthermore, PLGBM is able to constantly surpass the methods from [13] and [56] in all evaluated metrics. Summing these results with the approach taken by PLGBM, which is quicker to implement, they confirm that PLGBM, while being an overall worse choice than PlacERN for the record linkage task, may be a good alternative for a quick and competitive solution for the problem. The results also show that PLGBM is better able to handle class imbalance with fewer data points than PRF, since PRF lags behind PLGBM in $Pairs_{BR}$ but not so much in $Pairs_{US}$.

Finally, Table 2 shows that both previous approaches reach lower $F_{\beta=0.5}$ and normalized Gini coefficient values when compared to our preliminary models and PlacERN itself, with PE surpassing the heuristic approach of [13] in every evaluated metric. According to these precision-recall

curves for the $Pairs_{BR}$ dataset, the PE model from [56] has consistently lower metrics than all of our proposed solutions. Namely, we observe a difference of 0.151 in the normalized Gini coefficient when comparing PlacERN, the best-scoring model, with PE in the $Pairs_{BR}$ dataset, and 0.146 when comparing PLGBM to PE in the same set. Comparing $F_{\beta=0.5}$ further exacerbates this pattern.

It is important to note that the normalized Gini coefficient of PE in the $Pairs_{US}$ set is only 0.063 points below the best performer's value, with its $F_{\beta=0.5}$ score being 0.052 lower. Given that its performance on $Pairs_{BR}$ was lacklustre, this may indicate that the PE model, without the additional modifications proposed by its authors, is unable to handle class imbalance as well as the other models, and relies on a large dataset to be more effective. We acknowledge, however, that the full PEHAD model proposed by Yang *et al.* [56] displays a significant improvement over PE in their own work, so perhaps a full-blown implementation might have a better performance in this dataset than the other models.

We also note that PE is measured in its original work in regards to a different novel evaluation metric, under different scenarios. Meanwhile, the work from Dalvi *et al.* [13] is originally evaluated on a smaller, manually curated, and more balanced dataset. This might explain its poor performance on our evaluation.

Ablation Study In order to verify the impact of each encoder in the PlacERN model, we begin from a basic model consisting only of the word encoder (WE). Then, we gradually add other modules: the character encoder is added (WE + CHE), then the geographical encoder is added (WE + CHE + GE), and finally the category encoder is added, composing the final model PlacERN. The order of ablations follows the feature importance values obtained from studying our other models, as Section 6.6 shows. Each of these models has its parameters optimized by the same process applied to PlacERN. The results for this ablation study in all datasets are presented in Table 3.

From the results, we see that the full model, PlacERN, achieves the best $F_{\beta=0.5}$ and normalized Gini scores in both datasets. Each encoder adds to the model's performance, making the full model present an increase of 0.038 in the $F_{\beta=0.5}$ score and 0.029 in the normalized Gini coefficient when compared to WE only, in the $Pairs_{BR}$ dataset.

The most impact brought by the addition of an encoder may be seen in the WE + CHE version, where adding the character encoder boosts the $F_{\beta=0.5}$ by 0.023. These results are even less noticeable in the $Pairs_{US}$ set, our main supposition being the larger amount of training data. The fact that none of the encoders display a considerable impact on the score by itself points to the fact that the word encoder is a very powerful classification tool by itself, and each of the additions to the model serves to improve it in some edge cases.

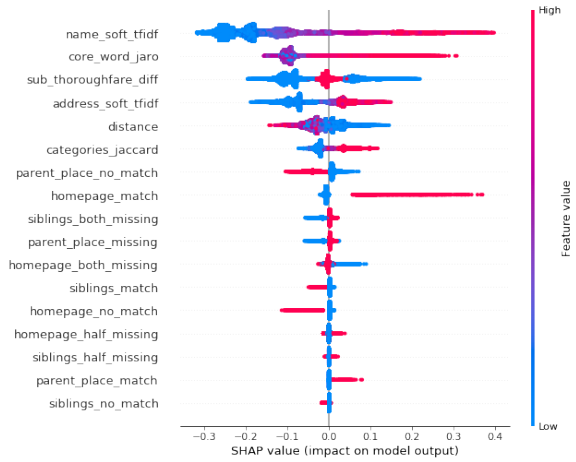
Table 3: Results of the ablation study for the PlacERN model.

Dataset	Model	Precision	Recall	$F_{\beta=0.5}$	Gini
$Pairs_{BR}$	WE	0.577	0.508	0.562	0.9
	WE + CHE	0.588	**0.573**	0.585	0.92
	WE + CHE + GE	0.6	0.557	0.591	0.921
	PlacERN	**0.617**	0.542	**0.600**	**0.929**
$Pairs_{US}$	WE	0.822	0.694	0.793	0.946
	WE + CHE	0.826	0.678	0.791	0.95
	WE + CHE + GE	**0.849**	0.668	0.805	0.956
	PlacERN	0.837	**0.712**	**0.809**	**0.959**

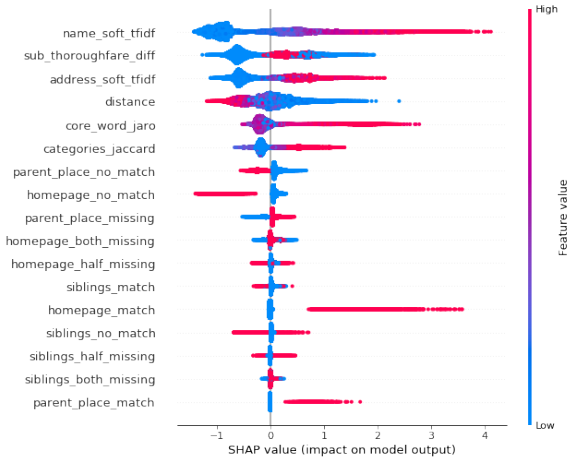
Feature Importance Study To study the importance of features in PRF and PLGBM, Shapley Additive Explanations [31] (SHAP) are conducted on their feature set. In this way, we expect to infer the impact of each place attribute and transfer this knowledge to the development of other models. SHAP is a game-theoretic approach which assigns an importance value to each feature for a particular prediction, and is able to aggregate those importance values to produce a general analysis on the effect of each feature in the model.

Figure 3 presents the SHAP summary plots for each of the features in the PRF (a) and PLGBM (b) models in the scope of validation data from $Pairs_{BR}$. The one-hot encoded features are broken up into their respective components categorical components. Each dot represents a sample, and a high absolute SHAP value indicates that a feature has more influence on the model classification, with dots in red representing higher feature values and dots in blue representing otherwise. In the Figure, features are ordered from top to bottom according to the sum of their absolute SHAP values for all samples. Note that the scale bar on the right refers to the color (feature value) of the predictions alone, and does not relate to the ordering of features in the y axis.

We see that features related to names and addresses have the most overall impact on the models. which in turn indicates that the most important information to detect replicated places are present in their names and addresses, with the distance between them being a good additive. The distance feature, as expected, is inversely correlated with positive samples, i.e. lower values push the model towards non-duplicates. The magnitude of the remaining features' influence on the model is much lower than their counterparts'. This analysis of the SHAP values leads us to focus the de-



(a) SHAP sumary for PRF.



(b) SHAP summary for PLGBM.

Fig. 3: Summary plots of SHAP values for PRF and PLGBM in $Pairs_{BR}$.

velopment of other models on place names, addresses and geographical coordinates, which is the approach taken in PlacERN.

Training and Execution Times To analyze the feasibility of fitting the models into a full record linkage pipeline, such as the one proposed in Section 4, we calculate the training and execution times for each model in Table 4, without accounting for pre-processing or post-processing steps. Analyzing training time is relevant to our use case because place databases are often dynamic, so recurrent training guarantees the quality of results over time. In a similar way, execution time matters for real-time use cases.

Whilst PlacERN is the best performer in both datasets in terms of classification scores, it takes 3149 seconds to train in the $Pairs_{BR}$ dataset. When compared to PLGBM, for instance, its training time is roughly 397 times higher.

Table 4: Training and execution times for all models.

dataset	Model	Training time (s)	Execution time (s)
$Pairs_{BR}$	Dalvi *et al.* (2014) [13]	7879.00	4.83
	PE - Yang *et al.* (2019) [56]	8598.00	2.97
	PRF	27.23	0.31
	PLGBM	**7.93**	**0.19**
	PlacERN	3149	80.83
$Pairs_{US}$	Dalvi *et al.* (2014) [13]	131610.00	25.44
	PE - Yang *et al.* (2019) [56]	130742.00	9.40
	PRF	364.26	1.56
	PLGBM	**32.05**	**1.28**
	PlacERN	542.00	232.42

Similarly, the training times of the model from [13] and the PE model stand out as being at least one order of magnitude higher than the other models', especially in the $Pairs_{US}$ dataset, where they reach upwards of 36 hours to train. It is relevant to note that most of the training time for PE comes from the embedding smoothing process, which is an active part of the model and relies on the construction of a places graph. This process is taken into account because, in a real scenario, these smoothed embeddings for each place would need to be re-generated to account for new entities and temporal changes in past ones. While training time does not disqualify a model for usage in a production environment, any researcher or engineer aiming to use it would have to dispense more resources to do so.

In regards to execution time, PLGBM takes only 0.19 seconds to classify the 60,925 test samples from $Pairs_{BR}$ and 1.28 seconds to classify the 307,017 test samples from $Pairs_{US}$, surpassing every other evaluated model. More noticeably, PlacERN takes longer than PLGBM, PRF, and each baseline method to execute, without using batch predictions. This points to the fact that PlacERN is more suited to a batch-wise or off-line case than a real-time one. These results also do not account for concurrency in real-time services, which may present itself as a bottleneck. In that regard, the LGBM library offers out-of-the-box concurrency and distribution support.

Discussion By considering the results presented in this section, one can conclude that PlacERN is able to effectively detect duplicate places, outperforming all baseline methods in all tested datasets. PLGBM also obtains better results than the baseline methods, and is extremely fast to train and execute, proving itself as a good alternative for solving the linkage problem in real-time services with latency constraints.

Furthermore, the experiments attest that the place's name and address information are the two most important fields

to extract relevant information from, in the entity matching task. Latitude and longitude values also serve as a good way to improve results. Following that, we find that the word encoder for names and addresses in PlacERN is already a powerful duplicate detection tool by itself. The additional encoders further increase the model's performance by small increments, making it easier to detect edge cases. Given the impact which classification errors in the record linkage task may cause in a database, each of these increments is a relevant addition.

7 Related Work

Works in the field of record linkage for place entities may be divided into two categories, based on how they approach the problem: (i) traditional approaches and (ii) deep learning approaches. This section discusses related works in these categories.

7.1 Traditional Approaches

The solution developed by Dalvi *et al.* [13] approaches the problem of detecting duplicate places by using information from place names and geo locations to build a name model and a spatial context model. The main concept behind the work is that places names are sufficiently represented from a set of core words, the remaining ones being background words. An Expectation-maximization algorithm is proposed to calculate a probability distribution of core words and a probability distribution of back-ground words. The background probability distribution is additionally calculated for each tile in a grid subdivision to account for spatial context. Finally, the work computes the probability of two places being duplicates by using the probability of the core word set from both of them being equal. This matching step is expanded by a dynamic programming algorithm to take string edit operations into account.

Another work [3] studies the problem of linking place records from different Location Based Systems (LBS) in a broad scope. The approach proposes a spatial blocking method to detect potential pairs, a similarity algorithm to match places, and a data fusion algorithm for the detected duplicates. The similarity algorithm uses a probabilistic approach to combine multiple traditional similarity metrics between place pairs to output a classification result.

Deng *et al.* [17] tackle the record linkage problem for two different data sets obtained from Chinese LBS. Similarly to [3], they use a multi-attribute matching strategy to classify duplicate places, having access to the name, address, geo location, and category of each place. The core of their work, on the other hand, resides in the combination strategy for these similarities, using an improved version of the Dempster–Shafer (D-S) evidence theory [16,46]. The textual similarity strategies proposed by their work are tailored towards Chinese places, and the category similarity assumes knowledge and manual fusing of the category schema from each source.

7.2 Deep Learning Approaches

The work of Yang *et al.* [56] draws inspiration from previous solutions for entity resolution and person re-identification tasks in different domains to create unsupervised representations of place entities from the Facebook Web service, noting that the user-facing nature of the service leads to place replications. They name this step as unsupervised feature generation, and utilize places' names, addresses, locations, and categories to do so.

This step utilizes FastText [4,33] to create name embeddings, using a *corpus* of 1.9 trillion words from public Facebook posts in the last 10 years, and a skip-gram Word2Vec [34] model trained on top of words from place addresses to create address word embeddings. Next, they incorporate category and geo location information by first applying a grid subdivision, then creating a places graph where places in the same grid or belonging to the same category are connected, and finally smoothing this graph by training a skip-gram objective function with negative sampling [41].

The results from this step are passed through an MLP network which uses a novel pairwise contrastive loss, an adapted version of a triplet loss. This model, named PE, is further improved by batch-wise hard sampling (PEH), a source-based attentive training strategy (PEHA) and cluster-based label denoising (PEHAD) [24]. The batch-wise hard-sampling uses a secondary distance metric, which is mentioned but apparently not explained in their work.

Their approach is similar to ours in many aspects, since our PlacERN model also leverages embeddings to build vector representations for places using the same attributes. On the other hand, our solution is built from the ground up with a focus on detecting duplicate places, and, as such, the network generates embeddings in a pairwise and supervised manner. Furthermore, the places graph generated during the embedding smoothing is costly to build and maintain, and usage of the text *corpus* of 1.9 trillion words is inaccessible to most researchers and companies alike, both due to its scale and its proprietary aspect.

Another work [44] tackles the similar problem of toponym matching. To better handle transliterations and semantic changes, they propose a deep architecture with two layers of siamese GRU networks processing character sequences for two place names, generating an embedding for each place, and merging them through a series of operations. This result is passed through a feed-forward network with

Dropout regularization [47] and a final sigmoid layer produces a probabilistic output indicating if the two toponyms match.

8 Conclusions and Future Work

We present PlacERN, a deep neural network to detect duplicate places, utilizing four different encoders to capture multi-level information about places and build an affinity between pairs. We further show how PlacERN is included in a working system by the proposal of a linkage pipeline. In addition to that, a set of pairwise features is engineered to train two supervised learning models, PRF and PLGBM. This feature set is studied in terms of its importance to drive the development of PlacERN. PlacERN is evaluated on top of two imbalanced industrial datasets, surpassing the performance of baseline methods. We consider relevant topics for future work to be the exploration of self-attention in the word and character encoders, and employing active learning to improve the proposed model due to the hardships of building a golden truth set.

9 Appendix: Hyperparameters

To improve reproducibility of our results, we describe in this appendix the optimization process utilized for the previous approaches, for our supervised models, and for PlacERN. We also provide the best hyperparameter values achieved during optimization. The value ranges used as input to the optimization algorithms may be found in [10].

9.1 Previous Approaches

We utilize Bayesian optimizations from Optuna [1] to tune the parameters of PE [56], and a grid search for the model of Dalvi *et al.* [13], both using validation data. The EM algorithm from [13] is executed for 10 iterations in $Pairs_{BR}$ and 5 iterations in $Pairs_{US}$, while PE [56] runs for 100 Optuna trials in $Pairs_{BR}$ and 50 trials in $Pairs_{US}$, using a median pruner after 5 warm-up trials and 3 warm-up steps. Geohashes are utilized as a means for the creation of tiles in both methods.

The optimal values obtained for the hyperparameters of [13] in the $Pairs_{BR}$ dataset are: $\lambda = 0.0$, Geohash precision $= 6$, $\alpha = 0.3$, and $t_c = 0.5$. For the $Pairs_{US}$: $\lambda = 0.0$, Geohash precision $= 6$, $\alpha = 0.9$, and $t_c = 0.5$.

Meanwhile, the embedding smoothing process from PE runs for 10 full epochs and uses 100,000 smoothing random walks with a fixed length of 10, a minimum frequency of 1, and a half window size of 5. The training batch size is fixed as 512 for $Pairs_{BR}$ and 1024 for $Pairs_{US}$, and the MLP uses 3 feedforward layers. The best values for the tuned hyperparameters of PE in the $Pairs_{BR}$ dataset are: $\alpha = 0.4$, smoothing negative sampling ratio $= 20$, neurons $= 512, 128, 128$, $t_c = 0.75$. For the $Pairs_{US}$ set: $\alpha = 0.5$, smoothing negative sampling ratio $= 5$, neurons $= 512, 128, 128$, $t_c = 0.75$.

9.2 Supervised Baseline Models

Both models (PRF and PLGBM) are tuned by Bayesian optimization, using Optuna for 100 trials in the scope of the $Pairs_{BR}$ and $Pairs_{US}$ validation data sets. The description of the hyperparameters for each model follows the parameter names from their respective libraries, with any value not shown assuming the default value.

PRF utilized the class_weight parameter set to *balanced* and the oob_score parameter set to *True* in both sets. The optimal tuned hyperparameter values for PRF in the scope of $Pairs_{BR}$ are: n_estimators $= 110$, max_features $= log2$, max_leaf_nodes $= 150$, min_samples_split $= 3$, $t_c = 0.9$. In the $Pairs_{US}$ set: n_estimators $= 150$, max_features $= sqrt$, max_leaf_nodes $= 150$, min_sample_splits $= 5$, $t_c = 0.9$.

The PLGBM model utilizes 10 early stopping rounds for 100 iterations in each trial, using a fixed class_weight value of *balanced*. The optimal values for its hyperparameters in $Pairs_{BR}$ are: lambda_l1 $= 1.247 \cdot 10^{-8}$, lambda_l2 $= 0.659$, num_leaves $= 95$, feature_fraction $= 0.5$, bagging_fraction $= 1.0$, bagging_freq $= 0$, min_child_samples $= 5$, $t_c = 0.5$. In the $Places_{US}$ set, the values are: lambda_l1 $= 1.132 \cdot 10^{-8}$, lambda_l2 $= 0.247$, num_leaves $= 256$, feature_fraction $= 0.62$, bagging_fraction $= 1.0$, bagging_freq $= 0$, min_child_samples $= 20$, $t_c = 0.5$

9.3 PlacERN

PlacERN is implemented with $L = 3$ feed-forward network layers, the first of them having $H_0 = 256$ neurons. Regarding the sequence lengths noted in Section 3, we use the 90th percentile of lengths for each field to extract $S^w = 5$, and $S^{ct} = 3$ for both sets, $S_n^{ch} = 42, S_a^{ch} = 32$ for $Pairs_{BR}$, and $S_n^{ch} = 26, S_a^{ch} = 23$ for $Pairs_{US}$. We use $B = 100$ distance buckets in the geographical encoder, and $m = 100$ dimensions for the embedding layers.

To use the pre-trained FastText embeddings in our model, their dimensionality is reduced to 100 beforehand by means of a Principal Component Analysis [39] dimensionality reduction script[8]. In order to improve reproducibility, we also fix the random seeds as 6810818.

The model is tuned by Bayesian optimization from Optuna for 50 trials on top of the $Pairs_{BR}$ validation data set

[8] https://github.com/facebookresearch/fastText/blob/master/reduce_model.py

and 20 trials in the $Pairs_{US}$ one, with a median pruner after 3 warm-up trials and 1 warm-up step. An additional early stopping callback with a patience of 2 epochs and a minimum change of 10^{-3} is also added as insurance against degenerate cases not detected by the median pruner. A batch size of 64 for $Pairs_{BR}$ and 1024 for $Pairs_{US}$ is utilized during training. The best tuned hyperparameter values for PlacERN and its ablated versions in both datasets are described in Table 5.

Table 5: Hyperparameters of PlacERN and its ablations, with the best values obtained in each data set.

Data set	Hyperparameter	PlacERN	*WE*	*WE+ CHE*	*WE+ CHE+ GE*
$Pairs_{BR}$	learning rate	0.001	0.005	0.001	0.002
	focal loss γ	1.5	2.0	2.0	0.0
	dropout rate	0.15	0.05	0.1	0.35
	GRU dimensions l	150	100	200	200
	CNN filters d	512	-	128	512
	CNN kernel size k	4	-	5	4
	t_c	0.5	0.5	0.5	0.4
$Pairs_{US}$	learning rate	0.001	0.001	0.001	0.001
	focal loss γ	2.0	2.0	1.0	1.5
	dropout rate	0.2	0.25	0.05	0.15
	GRU dimensions l	100	150	150	100
	CNN filters d	128	-	512	256
	CNN kernel size k	3	-	4	5
	t_c	0.55	0.55	0.65	0.6

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p. 2623–2631. Association for Computing Machinery, New York, NY, USA (2019). DOI 10.1145/3292500.3330701. URL https://doi.org/10.1145/3292500.3330701
2. Barbosa, L.: Learning representations of web entities for entity resolution. International Journal of Web Information Systems **15**(3), 346–256 (2018). DOI 10.1108/ijwis-07-2018-0059. URL http://dx.doi.org/10.1108/IJWIS-07-2018-0059
3. Berjawi, B.: Integration of heterogeneous data from multiple location-based services providers: a use case on tourist points of interest. Ph.D. thesis, Ecole doctorale d'informatique et mathématique de Lyon (2017)
4. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017)
5. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001). DOI 10.1023/a:1010933404324. URL http://dx.doi.org/10.1023/A:1010933404324
6. Buscaldi, D.: Toponym ambiguity in geographical information retrieval. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '09, p. 847. Association for Computing Machinery, New York, NY, USA (2009). DOI 10.1145/1571941.1572168. URL https://doi.org/10.1145/1571941.1572168
7. Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734. Association for Computational Linguistics, Doha, Qatar (2014). DOI 10.3115/v1/D14-1179. URL https://www.aclweb.org/anthology/D14-1179
8. Cohen, W.W., Ravikumar, P., Fienberg, S.E.: A comparison of string metrics for matching names and records. In: KDD Workshop on Data Cleaning and Object Consolidation. Association for Computing Machinery, Washington, DC (2003). URL https://www.cs.cmu.edu/afs/cs/Web/People/wcohen/postscript/kdd-2003-match-ws.pdf
9. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**, 2493–2537 (2011)
10. Cousseau, V.: A linkage pipeline for place records using multi-view encoders. Master's thesis, Universidade Federal de Pernambuco (UFPE), Pernambuco, Brazil (2020). URL https://github.com/vinimoraesrc/placern
11. Cousseau, V., Barbosa, L.: Industrial paper: Large-scale record linkage of web-based place entities. In: Anais Principais do XXXIV Simpósio Brasileiro de Banco de Dados, pp. 181–186. SBC, Porto Alegre, RS, Brasil (2019). DOI 10.5753/sbbd.2019.8820. URL https://sol.sbc.org.br/index.php/sbbd/article/view/8820
12. Cui, Y., Jia, M., Lin, T.Y., Song, Y., Belongie, S.J.: Class-balanced loss based on effective number of samples. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 9260–9269. Computer Vision Foundation / IEEE, Long Beach, California (2019)
13. Dalvi, N., Olteanu, M., Raghavan, M., Bohannon, P.: Deduplicating a places database. In: Proceedings of the 23rd International Conference on World Wide Web, WWW '14, p. 409–418. Association for Computing Machinery, New York, NY, USA (2014). DOI 10.1145/2566486.2568034. URL https://doi.org/10.1145/2566486.2568034
14. Damgaard, C., Weiner, J.: Describing inequality in plant size or fecundity. Ecology **81**, 1139–1142 (2000). DOI 10.2307/177185
15. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008). DOI 10.1145/1327452.1327492. URL https://doi.org/10.1145/1327452.1327492
16. Dempster, A.P.: Upper and lower probabilities induced by a multivalued mapping. The Annals of Mathematical Statistics **38**(2), 325–339 (1967). URL http://www.jstor.org/stable/2239146
17. Deng, Y., Luo, A., Liu, J., Wang, Y.: Point of interest matching between different geospatial datasets. ISPRS International Journal of Geo-Information **8**(10), 435 (2019). DOI 10.3390/ijgi8100435. URL http://dx.doi.org/10.3390/ijgi8100435
18. Dixon, P.M., Weiner, J., Mitchell-olds, T., Woodley, R.: Bootstrapping the gini coefficient of inequality. Ecology **68**, 1548–1551 (1987)

19. Dong, X.L.: Big data integration (2020)
20. Gini, C.: Variabilità e mutabilità: contributo allo studio delle distribuzioni e delle relazioni statistiche. [Fasc. I.]. Studi economico-giuridici pubblicati per cura della facoltà di Giurisprudenza della R. Università di Cagliari. Tipogr. di P. Cuppini, Cagliari, Italy (1912). URL https://books.google.com.br/books?id=fqjaBPMxB9kC
21. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. Journal of Machine Learning Research - Proceedings Track **9**, 249–256 (2010)
22. Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning word vectors for 157 languages. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), pp. 3483 – 3487. European Language Resources Association (ELRA), Miyazaki, Japan (2018). URL https://www.aclweb.org/anthology/L18-1550
23. Guo, J., Fan, Y., Ai, Q., Croft, W.B.: A deep relevance matching model for ad-hoc retrieval. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM '16, p. 55–64. Association for Computing Machinery, New York, NY, USA (2016). DOI 10.1145/2983323.2983769. URL https://doi.org/10.1145/2983323.2983769
24. Guo, X., Gao, L., Liu, X., Yin, J.: Improved deep embedded clustering with local structure preservation. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI'17, p. 1753–1759. AAAI Press, Melbourne, Australia (2017)
25. Hu, B., Lu, Z., Li, H., Chen, Q.: Convolutional neural network architectures for matching natural language sentences. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS'14, p. 2042–2050. MIT Press, Cambridge, MA, USA (2014)
26. Jiang, X., de Souza, E.N., Pesaranghader, A., Hu, B., Silver, D.L., Matwin, S.: Trajectorynet: An embedded gps trajectory representation for point-based classification using recurrent neural networks. In: Proceedings of the 27th Annual International Conference on Computer Science and Software Engineering, CASCON '17, p. 192–200. IBM Corp., USA (2017)
27. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: Lightgbm: A highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems 30, NIPS'17, p. 3149–3157. Curran Associates Inc., Red Hook, NY, USA (2017)
28. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1746–1751. Association for Computational Linguistics, Doha, Qatar (2014). DOI 10.3115/v1/D14-1181. URL https://www.aclweb.org/anthology/D14-1181
29. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Y. Bengio, Y. LeCun (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. ICLR, San Diego, California (2015). URL http://arxiv.org/abs/1412.6980
30. Lin, T., Goyal, P., Girshick, R.B., He, K., Dollár, P.: Focal loss for dense object detection. In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017, pp. 2999–3007. IEEE Computer Society, Venice, Italy (2017). DOI 10.1109/ICCV.2017.324. URL https://doi.org/10.1109/ICCV.2017.324
31. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (eds.) Advances in Neural Information Processing Systems 30, pp. 4765–4774. Curran Associates, Inc., Long Beach, California, USA (2017)
32. Marinho, A.: Approximate string matching and duplicate detection in the deep learning era. Master's thesis, Instituto Superior Técnico - Universidade de Lisboa, Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal (2018)
33. Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., Joulin, A.: Advances in pre-training distributed word representations. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), pp. 52 – 55. European Language Resources Association (ELRA), Miyazaki, Japan (2018). URL https://www.aclweb.org/anthology/L18-1008
34. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13, p. 3111–3119. Curran Associates Inc., Red Hook, NY, USA (2013)
35. Morton, G.: A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing. International Business Machines Company, Amonk, NY, USA (1966). URL https://books.google.com.br/books?id=9FFdHAAACAAJ
36. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML'10, p. 807–814. Omnipress, Madison, WI, USA (2010)
37. Ng, A.: Machine learning yearning: Technical strategy for ai engineers, in the era of deep learning (2019). URL https://www.deeplearning.ai/machine-learning-yearning/
38. Niemeyer, G.: geohash.org is public! (2008). URL https://web.archive.org/web/20080305102941/http://blog.labix.org/2008/02/26/geohashorg-is-public/
39. Pearson, K.: On lines and planes of closest fit to systems of points in space. Philosophical Magazine **2**, 559–572 (1901)
40. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
41. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, p. 701–710. Association for Computing Machinery, New York, NY, USA (2014). DOI 10.1145/2623330.2623732. URL https://doi.org/10.1145/2623330.2623732
42. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pp. 45–50. ELRA, Valletta, Malta (2010). http://is.muni.cz/publication/884893/en
43. Saito, T., Rehmsmeier, M.: The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. PloS one **10**, e0118432 (2015). DOI 10.1371/journal.pone.0118432
44. Santos, R., Murrieta-Flores, P., Calado, P., Martins, B.: Toponym matching through deep neural networks. International Journal of Geographical Information Science **32**, 1–25 (2017). DOI 10.1080/13658816.2017.1390119
45. Santos, R., Murrieta-Flores, P., Martins, B.: Learning to combine multiple string similarity metrics for effective toponym matching. International Journal of Digital Earth **11**(9), 913–938 (2017). DOI 10.1080/17538947.2017.1371253. URL http://dx.doi.org/10.1080/17538947.2017.1371253
46. Shafer, G.: A Mathematical Theory of Evidence. Princeton University Press, Princeton (1976)
47. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
48. Stefanidis, K., Efthymiou, V., Herschel, M., Christophides, V.: Entity resolution in the web of data. In: Proceedings of the 23rd

International Conference on World Wide Web, WWW '14 Companion, p. 203–204. Association for Computing Machinery, New York, NY, USA (2014). DOI 10.1145/2567948.2577263. URL https://doi.org/10.1145/2567948.2577263
49. pandas development team, T.: pandas-dev/pandas: Pandas (2020). DOI 10.5281/zenodo.3509134. URL https://doi.org/10.5281/zenodo.3509134
50. W3Techs: Usage statistics of structured data formats for websites (2020)
51. Wang, D., Zhang, J., Cao, W., Li, J., Zheng, Y.: When will you arrive? estimating travel time based on deep neural networks. In: AAAI, pp. 2500–2507. AAAI Press, New Orleans, LA, USA (2018)
52. Winkler, W.E.: String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In: Proceedings of the Section on Survey Research, Issues In Matching And Administrative Records Section, pp. 354–359. American Statistical Association, Alexandria, VA (1990)
53. Xiong, C., Zhong, V., Socher, R.: Dynamic coattention networks for question answering. In: 5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings. OpenReview.net, Toulon, France (2017). URL https://openreview.net/forum?id=rJeKjwvclx
54. Yalavarthi, V.K., Ke, X., Khan, A.: Select your questions wisely: For entity resolution with crowd errors. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17, p. 317–326. Association for Computing Machinery, New York, NY, USA (2017). DOI 10.1145/3132847.3132876. URL https://doi.org/10.1145/3132847.3132876
55. Yang, C., Bai, L., Zhang, C., Yuan, Q., Han, J.: Bridging collaborative filtering and semi-supervised learning: A neural approach for poi recommendation. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '17, p. 1245–1254. Association for Computing Machinery, New York, NY, USA (2017). DOI 10.1145/3097983.3098094. URL https://doi.org/10.1145/3097983.3098094
56. Yang, C., Hoang, D.H., Mikolov, T., Han, J.: Place deduplication with embeddings. In: The World Wide Web Conference, WWW '19, p. 3420–3426. Association for Computing Machinery, New York, NY, USA (2019). DOI 10.1145/3308558.3313456. URL https://doi.org/10.1145/3308558.3313456
57. Zaharia, M., Xin, R.S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M.J., Ghodsi, A., Gonzalez, J., Shenker, S., Stoica, I.: Apache spark: A unified engine for big data processing. Commun. ACM **59**(11), 56–65 (2016). DOI 10.1145/2934664. URL https://doi.org/10.1145/2934664