Experience: Enhancing Address Matching with Geocoding and Similarity Measure Selection

IOANNIS KOUMARELAS and AXEL KROSCHK, Hasso Plattner Institute, University of Potsdam, Germany
CLIFFORD MOSLEY, Concur, Bellevue, Washington
FELIX NAUMANN, Hasso Plattner Institute, University of Potsdam, Germany

Given a query record, record matching is the problem of finding database records that represent the same real-world object. In the easiest scenario, a database record is completely identical to the query. However, in most cases, problems do arise, for instance, as a result of data errors or data integrated from multiple sources or received from restrictive form fields. These problems are usually difficult, because they require a variety of actions, including field segmentation, decoding of values, and similarity comparisons, each requiring some domain knowledge.

In this article, we study the problem of matching records that contain address information, including attributes such as Street-address and City. To facilitate this matching process, we propose a domain-specific procedure to, first, enrich each record with a more complete representation of the address information through geocoding and reverse-geocoding and, second, to select the best similarity measure per each address attribute that will finally help the classifier to achieve the best f-measure. We report on our experience in selecting geocoding services and discovering similarity measures for a concrete but common industry use-case.

CCS Concepts: • **Information systems → Near-duplicate and plagiarism detection**; *Geographic information systems*; *Similarity measures*; *Clustering and classification*;

Additional Key Words and Phrases: Address matching, record linkage, duplicate detection, similarity measures, conditional functional dependencies, address normalization, address parsing, geocoding, geographic information systems, random forest

ACM Reference format:
Ioannis Koumarelas, Axel Kroschk, Clifford Mosley, and Felix Naumann. 2018. Experience: Enhancing Address Matching with Geocoding and Similarity Measure Selection. *J. Data and Information Quality* 10, 2, Article 8 (September 2018), 16 pages.
https://doi.org/10.1145/3232852

1 INTRODUCTION

The problem of *record matching*, which in the literature has been referred to by many different process names, including record linkage [2], duplicate detection [21], entity resolution [3], and

Authors' addresses: I. Koumarelas, A. Kroschk, and F. Naumann, Hasso Plattner Institute, University of Potsdam, Prof.-Dr.-Helmert-Strasse 2-3, 14482 Potsdam, Germany; emails: ioannis.koumarelas@hpi.de, axel.kroschk@student.hpi.de, felix.naumann@hpi.de; C. Mosley, Concur, 601 108th Ave NE Suite 1000, Bellevue, WA 98004, United States; email: cliff.mosley@concur.com.

https://doi.org/10.1145/3232852

data matching [4], is a well-studied problem [2, 7]. Two records match if they represent the same real-world object, such as a person, a product, or a company, despite their differences in their stored values. A typical method of classifying two records as duplicates is by determining the similarity or distance of their corresponding values. If the combined similarity is above a given threshold, then the record pair is considered to be a match and thus returned as a result.

In this article, we focus on matching records from a particular domain, i.e., records that contain address information. It is a notably difficult and important operation that allows for geo-spatial analysis in a number of areas, including health [23], traffic accidents [18], and natural disasters [10]. Address attributes, such as City or Street-address, appear frequently in many real-world applications and simultaneously have special (geographic) semantics that can be exploited to more effectively determine matching records. However, a system that is able to exploit these semantics must address several challenges.

Deduplicating addresses is particularly complex, when different countries or even areas within a country follow different *formats* and *rules*[1] for their addresses. Examples of such differences include building names, village and district names, house (building) and flat numbers, street types (street, highway, etc.), and more. Additionally, address data do not always follow the correct format, most commonly by mixing up the order of attributes or simply missing some values. On top of that, ambiguous abbreviations, slang, or synonyms make the problem even harder. For instance, "St" could abbreviate Street, Saint, State, Station, and so on.[2]

We employ and enhance *geocoding*, which is an operation that transforms a given address to Latitude and Longitude geolocation coordinates. Internally, systems that perform such an operation apply a mixture of techniques to obtain their result, including a special case of record matching against a clean address reference database. Such reference databases include qualitative location records that contain address information, such as Street-address and City, but also geolocation coordinates that are used to determine the final result of *geocoding*. These systems are usually described as *geocoding frameworks*, since *geocoding* is their main operation, and a popular example of such references databases is OpenStreetMap[3] (see Section 2 for more details). The best online providers of such systems usually enforce limits on the number of allowed queries, on their free version, and such queries are not always guaranteed to have some result, especially in cases of typographical errors in some of the query's tokens. Access to such services is usually provided with a REST interface, and there are tools providing cross-service access, such as the Python geocoder.[4]

Relevant literature is discussed in Section 2. We study the operations allowed by GIS systems and publicly available datasets for this reason, in Section 3. Then, we proceed with our process of finding the best similarity measure that is composed of three parts: First, in Section 4, preprocessing and enriching records' address information by applying geocoding and using Conditional Functional Dependencies with the help of a reference address database. Second, in Section 5, we propose two similarity measures, which serve as extensions of Monge-Elkan [19], and then we find the best similarity measure per attribute, among a number of typically used measures. Third, training and using a Random Forest (RF) classifier to determine whether the given similarities represent a duplicate or a non-duplicate pair of records, is explained in Section 6. Last, we conclude in Section 7 with our thoughts for future work, which includes modifications and extensions of this article's work.

[1] https://en.wikipedia.org/wiki/Address_(geography)#Mailing_address_format_by_country.
[2] See http://pe.usps.gov/text/pub28/28apc_002.htm for more examples.
[3] http://wiki.openstreetmap.org/.
[4] https://pypi.python.org/pypi/geocoder.

2 RELATED WORK

Record matching has been extensively studied over the years, under different names, as mentioned in the previous section. Several works include a thorough analysis of different methods that span across all the necessary steps to (1) normalize, (2) index and retrieve candidates of possibly matching records, (3) compare these retrieved candidates, (4) classify them as matches or not, and (5) evaluate the entire process [2, 7, 13]. In this article, we contribute to the first step, for which we apply *geocoding* and conditional functional dependencies (CFDs) to normalize the addresses of our records, and to the third step by experimentally analyzing the most commonly used similarity measures, and, finally, to the fourth step to choose a classifier to make the final decisions. For Steps (2) and (5) we employ standard techniques from the literature.

The authors of Reference [22] describe abstractly the process of creating a clean reference database, described as Geocoded National Address File for Australia (G-NAF), by merging smaller datasets from 13 organisations to be used for geocoding purposes on Australian addresses. The cost for such a project was estimated to cost around $12 million and ended up at $2 million due to improved technology. This still indicates how difficult and complex the procedure is, where even matching 70% of any given records is in many cases considered an acceptable result. Part of the process was also to resolve licensing problems when merging the smaller datasets into the single database.

Christen et al. [4] propose a Hidden Markov Model (HMM) that parses and separates an address into components. By matching these components using a rule-based engine, they identify the best matches from their reference database. Applying their methodology to a subset of G-NAF, they managed to match 94.94% of it on different levels (address, street, locality).

Finally, Miler et al. present a model for matching a traffic accident dataset against the OSM dataset, where the existence of Latitude and Longitude values is a vital part of the process, since the authors only consider pairs of records that are geospatially close [18]. In their application, they observed that many street names are based on persons' names. Thus, similarity measures that are suitable for people's names are suggested, and more specifically Jaro-Winkler [25], for which they found the best threshold to be 68%.

3 GEOCODING

Geographic Information Systems (GISs) commonly provide operations similar to and including *geocoding*. They used to have primarily the form of a desktop application, but nowadays they are typically offered as a service (XaaS). We first describe the typical set of provided operations. Most of the operators are based on some reference location datasets, which we describe thereafter.

3.1 Operations Provided by GIS Services

Operations that can be performed using these services include the following:

- *Normalization (or standardization)*: Format and clean components of addresses to comply with the country's postal service standards, including special character removal, lower- (or upper-) casing, zero padding number fields like ZIP-code, transposition of words, and encoding conversions.
- *Parsing*: Split addresses into its constituent components, including House-number, Street-name, ZIP-code, City, County-name, State-code (or State-name), and Country-code (or Country-name).
- *Verification*: Validate whether an address exists.
- *Geocoding*: Match a given query address to a geolocation of Latitudes and Longitudes, internally using records from clean reference datasets, such as the OSM, and by applying a mixture of operations, including record matching, interpolation, and more. The matched

geolocation is returned to the user. More often than not, these clean reference datasets contain entries with more information about a particular point. Thus, it can be also beneficial to include any kind of textual description along with or instead of the address. In our case, Hotel-name could be used as complementary information, which could lead to improved results.

- *Reverse (or inverse) geocoding*: For given geo-coordinates, return the nearest location in the reference data, such as an element record in OSM.

Example providers of these services, where the user sends his or her address as a REST request and the system sends back the geocoded coordinates, are ArcGIS,[5] Google,[6] and OpenStreetMap (OSM).[7] The latter is the only service based on open source data, available for research. Further open source implementations based on OSM include Nominatim,[8] Gisgraphy,[9] and Pelias.[10] Of these, the one that had the largest community support, based on github metrics (number of commits, stars and forks), and also seemed to produce the most reliable results in our case was *Nominatim*, and for this reason it was selected for our experiments. In Section 4, we explain how we use which operation.

3.2 Publicly Available Datasets of Location Information

The area of *geocoding* is very fragmented, where high-quality datasets only exist about smaller areas, and, in contrast, when larger areas are covered, quality usually drops. Therefore, we present the most complete, publicly available location datasets we could find. Since a large fraction of our use case's domain considers hotels in the U.S., we also include U.S.-only datasets, apart from the global ones.

- OpenStreetMap (OSM) is a worldwide volunteered geographic information (VGI) dataset, where people enter information in the form of primitives (nodes, ways, relations) that represent an element[11] and tags that contain metadata about the element.
- OpenAddressesIo[12] is "a global collection of address data sources, open and free to use," which was started by OSM users and is based mostly on government datasets.
- GeoNames[13] is another worldwide VGI dataset, which also allows for user modifications.
- TIGER US Census[14] is the acronym of topologically integrated geographic encoding and referencing system and is provided at a national level by the U.S. Census Bureau, including many geographic details, such as streets.
- Country Mappings[15] was our source of combinations of country codes (two- and three-character encoding) and the commonly used country's name.
- U.S.-specific: We collected further data from various public data sources.[16] While all files contained basic attributes, such as ZIP-code and State, they differed widely with regard to other attributes they provide and in their quality and coverage.

[5] https://www.arcgis.com/.
[6] https://developers.google.com/maps/documentation/geocoding/start.
[7] https://www.openstreetmap.org.
[8] https://nominatim.openstreetmap.org/.
[9] http://www.gisgraphy.com/.
[10] https://github.com/pelias/pelias.
[11] http://wiki.openstreetmap.org/wiki/Elements.
[12] https://openaddresses.io/.
[13] http://www.geonames.org/.
[14] https://www.census.gov/geo/maps-data/data/tiger-line.html.
[15] https://github.com/mledoze/countries.
[16] http://federalgovernmentzipcodes.us/, http://simplemaps.com/resources/us-cities-data, http://www.sqldbpros.com/2011/11/free-zip-code-city-county-state-csv/, http://www.unitedstateszipcodes.org/zip-code-database/, https://www.bls.gov/cew/cewedr10.htm.

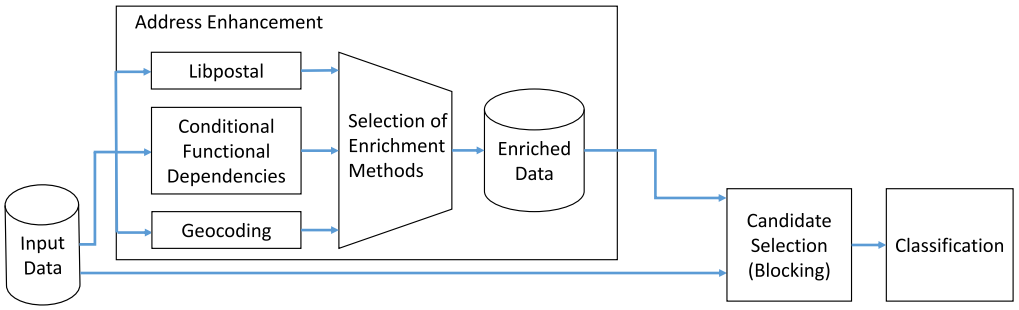


Fig. 1. Address enrichment process workflow.

To make use of the U.S.-specific datasets, we performed a merge operation to construct a single relation to include Latitude, Longitude, ZIP-code, City, County-name, State-code (and State-name), and Country-code (and Country-name). When pairs of records matched perfectly for all attributes except Latitude and Longitude, we merged the records and fused Latitude and Longitude to their average values.

4 ENRICHING ADDRESSES

Postal services of each country issue recommendations for address formats, which are not always followed in practice, which in turn leads to different representations of the same address. These formats change across local areas, such as cities, states, and so on, and more obviously across countries, which also use different languages. In addition, addresses may contain abbreviations, misspellings, mixed orders of attributes, or missing attributes. To handle the plethora of different formats and errors, we need to normalize and parse the addresses to ultimately compare them on a per-attribute level.

In this section, we, first, define the different processes that we used to enrich addresses; second, we describe our use-case dataset about hotels; and, finally, we evaluate the enrichment processes on the dataset. This process can be seen in Figure 1, where we show the executions over the original data and the enriched ones. The latter are a combination of the three proposed enrichment processes. Having either of the two datasets, in real-world applications we then perform the candidate selection, using blocking, to select the most promising candidate pairs for the next step, which in our case is classification. Keep in mind that in our experiments, we perform the candidate selection before any enrichment takes place, on the original data to ensure that the same set of candidates is selected with or without enrichment.

4.1 Enrichment Processes

We describe three different processes described to enrich addresses. Each can be used in isolation or in conjunction, as we show in Section 4.3.

4.1.1 Parsing. Since address parsing, i.e., labeling the individual parts of an address, is a difficult task in its own right,[17] we decided to use the popular address *parser* and *normalizer* Libpostal.[18] Libpostal claims to have an accuracy of 99.45% and support for over 60 languages.

Libpostal first *expands* the address, eliminating abbreviations and converting numbers to a uniform representation. In the second step, each part of the address is labeled according to its

[17] https://www.mjt.me.uk/posts/falsehoods-programmers-believe-about-addresses/.
[18] https://mapzen.com/blog/inside-libpostal/.

meaning. As an example, the address "1317 e hwy sixty seven decatur 35601 alabama" would be normalized to "1317 east highway 67 decatur 35601 alabama" and afterwards parsed to "House-number: 1317, Road: east highway 67, City: decatur, ZIP-code: 35601, State-name: alabama". Thus, the parsing step can also be used to infer attribute values, which might exist in wrong positions; for instance, the Street-address may also contain other attributes, such as the City.

4.1.2 Enrichment through Conditional Functional Dependencies. Filling missing attributes is an important task, which apart from using Libpostal's parsing, may be facilitated using other approaches. If there exist sufficient attributes with values, then we can use functional dependencies (FDs) to infer some of the missing values.

As an example, consider the dependency between a ZIP-code and a City. If we know that the FD ZIP-code→City holds, and we know the ZIP-code of a record, then we can infer the City by consulting a reference address database (such as those described in Section 3.2). Nonetheless, even for this example of ZIP-codes, the problem might be more complicated, since a ZIP-code can be either inside one City or span over a region of multiple Cities (that can be in multiple Counties or States), commonly found in military bases, and the same ZIP-code can exist in many countries across the globe.

General FDs are only valid if they apply across the entire dataset. CFDs, however, are FDs that need to apply only to a subset of the records, as specified by some condition. As shown in Reference [14], partial enrichment with CFDs is easy to apply, simple, and better than other methods. Moreover, it improves the results on its own and enhances the address normalization step (Section 4.1.3). For instance, the functional dependency ZIP-code→City might be true only for a specific country. To make use of this concept, for each incoming record, we examine the present non-null values. For each combination of these values, we check with the reference database whether we can uniquely determine a value for one of the missing attribute values.

Consider a record with (98103, us) as ZIP-code and Country-code but a missing value for City. We collect all records with the given values (98103, us) and count the number of distinct values for City. There are three possible outcomes: (i) If the original count is 1, then we were successful and can infer the right hand side (RHS) attribute value. There is only one single city for the combination (98103, us), namely seattle. (ii) If the count is 0, then we cannot directly infer a missing value. Thus, we relax this left hand side (LHS) condition by considering fewer attributes to infer the missing attribute value. In this case, we recursively count the number of distinct cities for the ZIP-code 98103 only and the number of distinct cities for the Country-name us only. (iii) If the count is higher than 1, then we cannot uniquely infer the city, and there is no reason in examining more relaxed LHS conditions, as those will certainly have more results. Once a single value has been determined, we iteratively use this new value in the LHS to create further, more complex combinations, to check until no new values can be filled in.

For lack of a high-quality reference dataset, we implemented this process for U.S. addresses only and were able to infer information across the address attributes of ZIP-code, City, County-name, State-code, State-name, Country-code (two- or three-character format), and Country-name (common and formal names). In principle, the same procedure could be followed at a global level, for instance, by parsing the entire OSM dataset and producing a relation with the above information.

4.1.3 Enrichment through Geocoding. The previous steps could be used in isolation, but should also enable us to be more successful during the geocoding process. As previously mentioned, the goal of this step is to match the query's information with a record from a clean reference database (described in Section 3.2) to obtain its geolocation. Afterwards, by performing *reverse geocoding* we obtain a proper, formal representation of the record. In fact, most geo-coding systems return

Table 1. Selecting the Best Enrichment Process, Based on the Gold Standard of 240,944 Records and 301,155 Pairs

	% records	% pairs	% pairs with 0 distance
Nominatim	24.29	16.62	73.83
Libpostal + Nominatim	20.68	14.15	75.70
CFD + Nominatim	37.21	29.07	78.73
CFD + Libpostal + Nominatim	33.42	26.88	80.17
Ensemble	43.65	34.89	77.31

the reference record already during the *geocoding* step. The actual process of geocoding is more difficult than what has been already described in Section 2, including house number interpolation, in case the requested address is not available but neighboring house numbers are, synonyms in other languages, font encoding problems when not everything is in UTF-16 (usually UTF-8 is used), and more. We chose to use the open source system Nominatim, which we set up as a local server with an Intel Xeon CPU, 3.07GHz, eight cores, a RAM of 24GB and a HDD of 2TB.

4.2 Use-case: Hotel Dataset

To evaluate the different approaches for geocoding and matching datasets, a gold standard is needed. The problem with publicly available datasets, such as North Carolina Voter Registration,[19] is that they do not contain significant variations among duplicates—as their quality is typically better than what we observe in our real-world queries. Therefore, we proceeded to use a real-world dataset that our industry partner Concur[20] provided, which includes information of *hotels* around the world, along with a gold standard of duplicate pairs. This dataset includes 364,965 records with 36 attributes and a size of 91.2MB. Along with the records, Concur reported 384,238 duplicate pairs, i.e., pairs of records that have been verified to represent the same hotel. Keep in mind that clusters of multiple records representing one hotel lead to many duplicate pairs. In fact, the largest cluster contained 54 records (accounting for 1,431 pairs), and 124,653 records had no annotated duplicates. The hotel dataset includes the attributes Hotel-name, Street-address, ZIP-code, City, County-name, State-code, Country-code, Latitude, and Longitude, which all contain real-world, human-made errors and misspellings.

4.3 Experimental Evaluation

Geocoding is the most important of the processes, because it enriches records with the most reliable address information. We used the first two of the processes as pre-enhancing steps for the last step, aiming for better geocoding results. In this section, we report how many hotel records are in fact matched and enriched by these combinations of the processes. We focused on the eight countries (US, FR, DE, GB, IT, CA, CN, AT) that represent the largest fraction of our records, keeping 240,944 records and 301,155 pairs of the gold standard.

By examining Table 1, we can observe, first, the impact on the percentage of records that have been enriched; second, the percentage of pairs from the gold standard for which *both* of the records have been enriched; and, finally, from this the percentage of records where the geolocation distance, based on latitude and longitude, is zero. These statistics help us to make the decision among the first four choices that include combined enrichment processes, each line representing a set of

[19] http://dl.ncsbe.gov/index.html?prefix=data/.
[20] https://www.concur.com/.

consecutive steps. We chose the combination CFD + Nominatim for its simplicity and good results. A brief experimental comparison with Ensemble can be seen in Section 6.2.

Regarding the execution time per record, the mean values that we observed under a multi-threaded (Java 8 parallelStream) execution, were 17ms for CFDs, 96ms for Libpostal, and 4,057ms for Nominatim. In total, the execution time of the classification process, not including the enrichment, was increased from 1,109s to 1,329s (+220s) due to the added information contained in the attributes and, consequently, in the similarities. For performing the GET requests on the latter system, multiple Java libraries were tested, and the execution times were always at similar levels. The final Java library that was used is unirest,[21] and we set the timeout of a request to 20s. We assume that Nominatim's long execution time may happen because of long system retrieval times, e.g., for frequently occurring tokens. Let us now examine a few experiences in using those systems:

4.3.1 Libpostal. By using Libpostal before calling Nominatim we try to achieve two things:
(1) Fill missing values that might have been contained in other attributes. By *parsing* a concatenated string where we join all the information about the address attributes that we know about, we expect to improve our quality of attributes. However, several issues are faced here:

- Improper categorization of tokens. A ZIP-code could be identified as a House-number and vice versa, a Street-name as a City, and so on. As an example, the result of the input "`7210 ga hwy 21 31407 port wentwort ga us`", incorrectly (and inexplicably) returns `31407` as the House-number and `7210` as the ZIP-code, swapping their respective categories.
- Non-identification of certain tokens leads to the concatenation of tokens. Given the input "`4900 bryant irvin rd 76132 3616 fort worth tx us`", Libpostal returns the House-number `49003616`.

These two problems appeared in more than 50% of the cases in total. This made the *parsing* part of this library unreliable and was thus avoided.
(2) *Expand* and normalize the address query that will be given to Nominatim to eliminate abbreviations (i.e., `St` is converted to `Street`), represent numbers in a uniform format (i.e., `IX` is converted to 9), and correctly position the tokens to reflect the correct pattern (i.e., house-number in front, then street-name, etc.). This should increase the success rate of geocoding. Unfortunately, Libpostal returns multiple, unranked expansions as shown in Table 2. We select the largest expansion.

4.3.2 CFD. The goal as stated in Section 4.1.2 is to enrich the record with information that can only coexist with our existing attributes' values. As an example in Table 3, we are able to obtain the City, State, and the County-name as well. Overall, our entire process was able to enhance $\frac{131,391}{135,583} = 96.90\%$ of the US records with at least one additional attribute value. Using Nominatim alone yielded only 58,529 enhanced records (43.16%); using CFD + Nominatim led to 89,670 (66.13%) geocoded records.

4.3.3 Geocoding. When geocoding is successful, we apply reverse geocoding to obtain proper address details, as explained in the beginning of Section 4.1.3. The address information that is returned contains important additions and is useful for our later matching process. As an example, consider Table 4, where by providing the query of the original record to Nominatim, the returned result contains a lot more address attributes, which clarifies better the location of the hotel.

[21] http://unirest.io/java.html.

Table 2. Libpostal Unreliable Expansions

"507 e main st 99328 dayton wa us"
"507 e main street 99328 dayton wa us"
"507 e main street 99328 dayton western australia us"
"507 e main street 99328 dayton washington us"
"507 e main saint 99328 dayton wa us"
"507 e main saint 99328 dayton western australia us"
"507 e main saint 99328 dayton washington us"
"507 east main street 99328 dayton wa us"
"507 east main saint 99328 dayton wa us"
"507 east main saint 99328 dayton western australia us"
"507 east main saint 99328 dayton washington us"

Table 3. CFDs Example

attribute	original value	enriched value
City		marshal
ZIP-code	56258	56258
County-name		lyon
State-code	mn	mn
State-name		minnesota
Country-code	us	us
Country-name		united states

Table 4. *Geocoding* (Nominatim) Example

attribute	1. original record	2. *geocoded* to	3. *reverse geocoded* to
Latitude		42.6334525	42.6334525
Longitude		−88.6371988	−88.6371988
Street-address	511 e walworth ave		511 e walworth ave
ZIP-code	53115		53115
City	delevan		delevan
County-name			walworth county
State-code	wi		wi
State-name			wisconsin
Country-code	us		us
Country-name			united states
Query	511 e walworth ave 53115 delavan wi us		
Display-name			walworth avenue, delavan, walworth county, wisconsin, 53115, united states of america

5 CHOOSING BEST SIMILARITY MEASURES

Address information, which is the focus of this study, feature a wide variety of attributes and attribute types. The goal of this section is to recommend specific similarity measures for common attributes of addresses. To this end, we first introduce popular measures and then evaluate each of them for each of the attributes, recommending a best measure for each.

5.1 Similarity Measures

A similarity measure is a function that takes two input values, $v1$ and $v2$, and returns a value between 0.0 and 1.0, with 0.0 meaning they are completely different and 1.0 meaning they are exactly the same: $sim(v1, v2) \rightarrow [0.0, 1.0]$.

Research and industry have established a wide array of various similarity measures for many different value types and purposes. In many cases, string-based measures are originally defined as distance measures, whose values are converted to a similarity by subtracting the normalized distance from 1. We briefly review different categories of distance/similarity measures:

Edit-based. These string-based measures consider the whole two strings and count the edit operations (e.g., insertions, deletions, replacements or swaps of characters) that are needed for one input to be converted to the other. We evaluated the following edit-based similarity measures:

- Exact Match (EM) returns 1.0 if the strings are exactly the same, and 0.0 otherwise.
- Hamming (HM) counts the positions at which characters are different [12].
- Levenshtein (LS) counts the minimum number of edit operations to transform one string to the other [17].
- Damerau-Levenshtein (DL) expands the Levenshtein measure with the transposition operation for two adjacent characters (a common typo) [6].
- Jaro-Winkler (JW): An extension of the Jaro distance [16], which counts the transpositions of characters needed to transform one string to the other, as long as they happen within less than half the string's length. JW then favors cases where the two input strings have a common prefix [25].
- LongestCommonSubsequence (LCS) is the longest sequence of characters that is common in the two strings [5], in contrast to LongestCommonSubstring [11], which demands the characters to be adjacent.

Token-based. These measures tokenize the strings and compare the sets of tokens:

- Jaccard (JC), also known as Jaccard index or Jaccard similarity coefficient [15], represents how many tokens the two strings share [19].
- Jaccard n-gram (JCN) is a variation of JC where the tokens are produced using a sliding window of size n (i.e., n-grams).

Hybrid. Hybrid measures combine the advantages of both previous categories by using an internal edit-based similarity and applying it to combinations of tokens of the entire input values. The final similarity is then the average similarity of matched pairs.

- Monge-Elkan (ME) goes through all tokens of the first input string and finds the token of the second string with the maximum similarity, which can also be re-used for further matches [19]. ME returns the mean similarity of the matched token pairs.
- Monge-Elkan Greedy Symmetric (MEG) modifies ME to avoid the re-use of matched tokens, giving both input values the same importance. We calculate the similarity of each token pair using the internal similarity measure and then greedily choose best matching pairs.
- Stable-Matching (SM) ensures that no token is more similar to another token than to the one it is matched to, while simultaneously that other token is also more similar than to its own match. We use the Gale Shapley algorithm [8] to find the matching. As the algorithm is not symmetric, we repeat it with swapped input values and use the average similarity as the final result.

We combined the hybrid measures with three different internal similarity measures, namely Levenshtein, DamerauLevenshtein, and JaroWinkler, abbreviated as ME-LS, ME-DL, and so on.

For the purpose of duplicate detection we define a *threshold* for each similarity measure and attribute: If the similarity is above the threshold, then we call it a *match*; if it is below, then it is a *non-match*. Having defined the different similarity measures, we proceed in the next section to identify what is the best mapping between them and the common address attributes and which threshold to choose for each combination.

5.2 Experimental Evaluation

We selected the seven attributes described in Section 4.2, which are universal in most countries. In this section, we first describe the process to determine precision, recall, and f-measure. Second, we evaluate all the previously defined similarity measures across our selected record's attributes and select combinations with the highest potential in terms of some metric, such as f-measure or execution time. For all experiments, we perform a 10-fold cross-validation on our dataset and report the mean scores.

5.2.1 Selecting Duplicate and Non-Duplicate Pairs. The gold standard that is given to us by our industry partner, contains 130,428 pairs of records that are marked as *duplicates (DPL)*. Since we want to train a classifier for duplicate detection, we need to calculate the f-measure, based on precision and recall values, which in turn needs knowledge about false positives (FP) that are returned in a detection run. Identifying FPs needs *non-duplicate (NDPL)* pairs, and for this reason we created a process that produces them, in the same spirit as in Reference [1]. Since we already own a DPL set, we follow a different strategy, based only on Blocking. It would be easy to choose many trivial NDPL pairs, simply by choosing many very dissimilar records. For a more realistic evaluation, we describe how we generated these difficult NDPLs:

(1) We insert all DPL into a UnionFind (UF) [9] data structure, which provides us with transitive closure functionality. UF forms transitive closure groups, which include all different representations (records) of the same entity that are connected in the gold standard through direct and indirect (transitive) connections.
(2) We use the Blocking technique to partition the records and thus save comparisons and reduce the number of trivial comparisons. We apply blocking individually to the attributes Hotel-name, Street-address, City, and Zip-code, using the entire value as the blocking key. These attributes have a high uniqueness, which means that the same value is not common across many records. Each bucket represents all records which have the bucket's value for that specific attribute. Finally, we go through all these attributes and their respective buckets, fetch all record combinations, and add them to a collective set of pairs that contains both DPLs and NDPLs.
(3) We remove the pairs that belong in the same transitive closure group, i.e., they are duplicates, using the UF structure.

The two sets of pairs, DPL and NDPL, constitute our full gold standard (*Full-GS*), which we use for evaluation in the following steps.

5.2.2 Best Similarity Measure per Attribute. To determine the best similarity measure for each attribute, we evaluate for each one all similarity measures and calculate the f-measure for 100 different thresholds ($[0.01, 0.02, \ldots, 1.0]$). We thus now know for each similarity measure the best f-measure, along with the corresponding thresholds and the overall execution time. We report the thresholds for which optimal results were achieved but do not apply them in the classification step. We leave the selection of feature thresholds to the Random Forest classifier. With this information, users can select the similarity measure that maximizes the f-measure, minimizes the execution time, or something in between. In our case, we solely target maximizing the f-measure. We discuss some exemplary results as follows:

- Hotel-name: Figure 2(a) shows JCN to achieve the highest f-measure, while at the same time being one of the fastest measures as well. The best f-measure was achieved for the threshold of `0.29`.

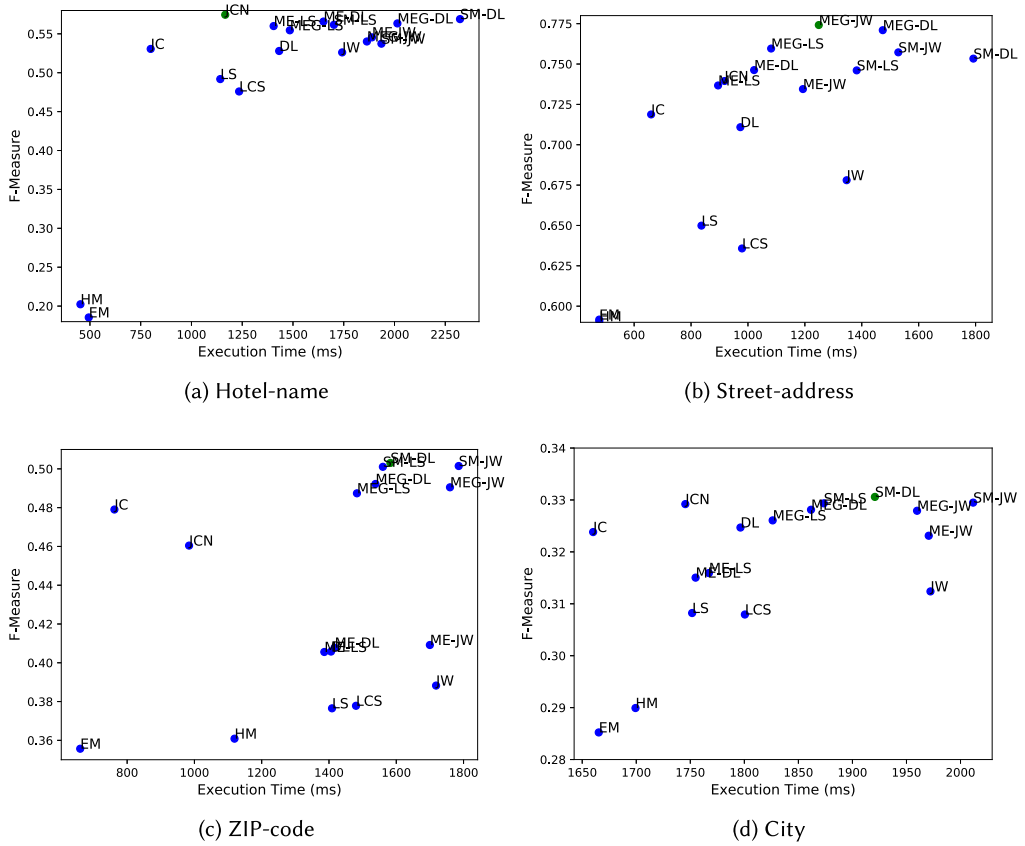


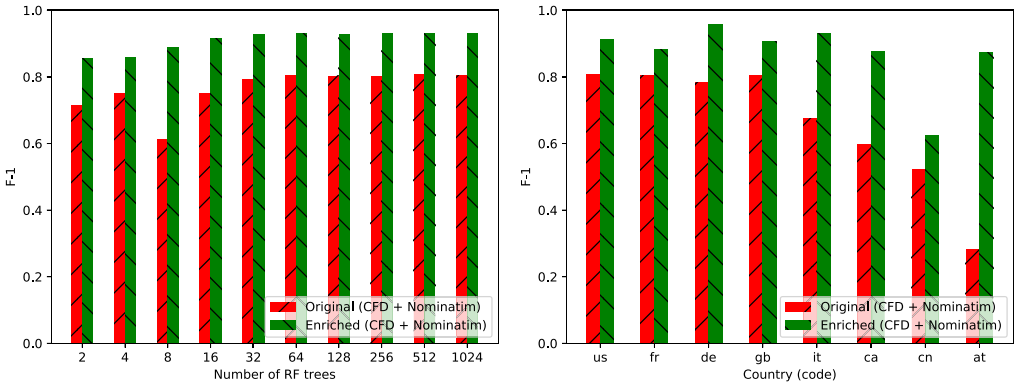
Fig. 2. F-Measures for similarity measures per attribute.

- Street-address: MEG-JW is shown in Figure 2(b) to have the best f-measure, with the threshold of `0.95`. A Street-address is usually comprised of multiple tokens, explaining the good performance of this token-based measure.
- ZIP-code: The best choice as similarity measure is the SM-DL, achieved at the threshold of `0.9`, as shown in Figure 2(c), since there are ZIP-codes that contain multiple tokens.
- City: Figure 2(d) shows that the best choice as similarity measure is the SM-DL, with `0.46` as the best performing threshold.
- County-name: For space reasons, we do not show results for the following choices. For countries, it is the same as ZIP-code's and City's – SM-DL, with the threshold of `1.0`.
- State-code and Country-code: We use the EM similarity measure.
- Latitude and Longitude: We use the well-established Haversine distance.[22]

6 CLASSIFICATION FOR DUPLICATE DETECTION

Having enriched records and knowing the best similarity measure per attribute, from Sections 4 and 5, this section introduces a classifier to detect which of these pairs are duplicates or not. As features, we provide the calculated similarities, for every pair of the *Full-GS* (see Section 5.2.1), with

[22] https://en.wikipedia.org/wiki/Haversine_formula.



(a) F-Measure across number of RF trees

(b) F-Measure across countries

Fig. 3. Effect of Random Forest on F-Measure under different configurations.

the label 1 representing duplication and `0` the opposite. To this end, we have chosen Random Forest (RF) as the classifier for our experiments[23] for its scalability, ease in parametrization, and out-of-the-box applicability to most problems. This makes it easier to deploy in real-world applications. We have also experimented with Support Vector Machines, which needed excessive memory while performing only slightly better on our samples. An additional approach we have tried was to use a threshold-based classifier, where each attribute's similarity is weighted, and if it passes a given threshold, then the pair is a match. We employed a Genetic Optimizer, since the problem of maximizing f-measure is non-convex [20], to find the optimal weights and threshold but achieved only much lower f-measures. Another industry-relevant reason to choose RF is its ability to explain specific outcomes. Finally, we did not invest more time trying to resolve the above issues on the other classifiers, as it is not the main contribution of our article.

In the next sections, we describe how to first *train* it and second *evaluate* it on our domain's dataset.

6.1 Best Parameter for Random Forest Classifier

The RF classifier has a number of different parameters that can be configured, with the number of *trees* being typically the most important parameter to be examined, which we also do in our experiments. Further parameters include the number of *attributes* to be sampled on at each node and the criterion to make the binary *splits* in each node. For both these parameters, as well as all remaining ones, we keep to the library's default values,[24] which are typically used in practice [24]; for the former, we use $\sqrt{m}$, with m being the number of attributes, and for the latter we use the GINI index. As the best-performing enrichment process is *CFD + Nominatim*, as explained in Section 4.3, we decided to perform experiments w.r.t. the number of *trees*. Second, we verify the parameter's selection by performing experiments across countries.

In Figure 3(a), we can see the differences in f-measure for the subset of records that have been enriched by *CFD + Nominatim* and in their original state. In other words, it helps us understand whether the enrichment helps (it does), and at the same time it shows us that for 64 trees we have

[23]We used the Java library from https://haifengl.github.io/smile/.

[24]https://github.com/haifengl/smile/blob/master/core/src/main/java/smile/classification/RandomForest.java.

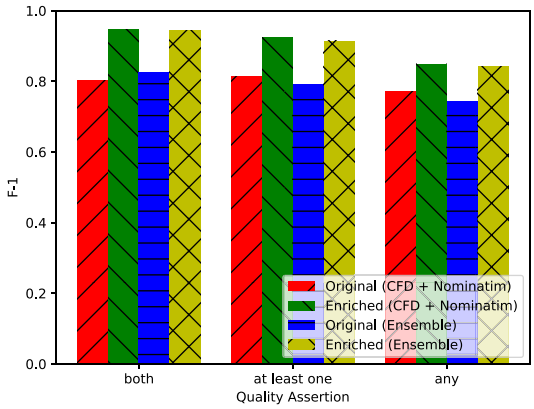


Fig. 4. F-Measure across GS usages.

a good compromise between the two data states, keeping in mind that also in future applications we cannot expect all records to be enriched. We continue our experiments with 64 trees.

In Figure 3(b), we can observe the effect of enrichment across the eight countries that were selected in Section 4.3. In "US" both *CFDs* and *Nominatim* contribute to the result, whereas in the rest countries only Nominatim can enrich the records. Overall, we observe that in all countries the effect of the *CFD + Nominatim* enrichment is positive.

6.2 Evaluation with Quality Assertion

In search of understanding the importance of the enrichment, we performed three experiments by controlling the quality of the used record pairs. We created three datasets in which pairs from Full-GS were "both" enriched, "at least one" was enriched, or "any" on enrichment. Based on these three configurations, we can aim for a good intuition on how much enrichment contributes to the final matching quality. The previous experiments took place using the middle solution ("at least one") as the default configuration.

In Figure 4, we can see the difference among these three configurations with two things being clear. First, the quality is better in the configurations where one or both of the records are enriched, which means that our hypothesis that enrichment does contribute holds true. Second, the cases that were enriched and performed better than the original were higher-quality ones, as the original records in the two left configurations have a better f-measure. Last, the improvement of using the *CFD + Nominatim* process is profound, although *Ensemble*, while being more expensive, manages to affect more records while achieving almost the same f-measure or even better.

7 CONCLUSION

In this work, we showed a recipe for matching records when address information is present, in which case we can achieve a higher match rate compared to other types of attributes. More specifically, we showed that by using different address enrichment processes, and by finding the best similarity measure for each address attribute, we were able to improve the f-measure results of the consequent Random Forest classifier. Regarding the address enrichment processes, Conditional Functional Dependencies along with a Geocoding framework, Nominatim in our case, achieved the best results. For most of our attributes, hybrid similarity measures performed the best. In particular, for Hotel-name Jaccard n-gram performs the best under the threshold of `0.29`, whereas for Street-address our improved version of Monge-Elkan with Jaro-Winkler as the token

similarity measure was the best using the threshold of `0.95`. Finally, for the attributes ZIP-code, City, and County-name, Stable Matching with Damerau-Levenshtein as the token similarity measure performed the best, with thresholds of `0.9`, `0.46`, and `1.0`, respectively.

Several avenues of future work emerge: In the future, we plan to experiment with more enrichment techniques, in the same principle as the CFDs. Also, as reference databases grow ever larger (the complete OSM is already 803GB), a better coverage can be achieved, but matching methods need to be distributed among multiple machines. Third, new geocoding frameworks are continuously emerging, so experimenting with them is a necessary part on all address-related tasks. Finally, achieving all of the above in lower execution times demands for some distributed environment, such as Apache Spark,[25] which is another interesting direction, where experimentation with distributed indexes, caches, and more could help.

REFERENCES

[1] Peter Christen. 2007. A two-step classification approach to unsupervised record linkage. In *Proceedings of the 6th Australasian Conference on Data Mining and Analytics, Volume 70*. Australian Computer Society, Inc., 111–119.

[2] Peter Christen. 2012. *Data Matching—Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer Data-Centric Systems and Applications.

[3] Peter Christen, Ross Gayler, and David Hawking. 2009. Similarity-aware indexing for real-time entity resolution. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM'09)*. ACM Press, 1565–1568.

[4] Peter Christen, Alan Willmore, and Tim Churches. 2006. Data mining. Springer-Verlag, Berlin, 130–145.

[5] Vaclav Chvatal and David Sankoff. 1975. *Longest Common Subsequences of Two Random Sequences*. Technical Report. Stanford University.

[6] Fred J. Damerau. 1964. A technique for computer detection and correction of spelling errors. *Commun. ACM* 7, 3 (1964), 171–176.

[7] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. 2007. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.* 19, 1 (2007), 1–16.

[8] David Gale and Lloyd S. Shapley. 1962. College admissions and the stability of marriage. *Am. Math. Month.* 69, 1 (1962), 9–15.

[9] Bernard A. Galler and Michael J. Fisher. 1964. An improved equivalence algorithm. *Commun. ACM* 7, 5 (1964), 301–303.

[10] Debarati Guha-Sapir, Rhonda Davis, Melanie Gall, Pascaline Wallemacq, and Susan Cutter. 2015. Exploring the potential of geocoding the impact of disasters: The experience of global and national databases. In *EGU General Assembly Conference Abstracts*, Vol. 17.

[11] Dan Gusfield. 1997. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press.

[12] R. W. Hamming. 1950. Error detecting and error correcting codes. *Bell Syst. Techn. J.* 29, 2 (1950), 147–160.

[13] Thomas N. Herzog, Fritz J. Scheuren, and William E. Winkler. 2007. *Data Quality and Record Linkage Techniques* (1st ed.). Springer.

[14] Ihab F. Ilyas, Xu Chu, and others. 2015. Trends in cleaning relational data: Consistency and deduplication. *Found Trends Databases* 5, 4 (2015), 281–393.

[15] Paul Jaccard. 1901. Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines. *Bull. Soc. Vaud. Sci. Natur.* 37 (1901), 241–272.

[16] Matthew A. Jaro. 1989. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *J. Am. Statist. Assoc.* 84, 406 (1989), 414–420.

[17] Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Sov. Phys. Dokl.* 10, 8 (1966), 707–710.

[18] Mario Miler, Filip Todić, and Marko Ševrović. 2016. Extracting accurate location information from a highly inaccurate traffic accident dataset: A methodology based on a string matching technique. *Transport. Res. C: Emerg. Technol.* 68 (2016), 185–193.

[19] Alvaro E. Monge and Charles P. Elkan. 1996. The field matching problem: Algorithms and applications. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD'96)*. 267–270.

[25] https://spark.apache.org/.

[20] Ye Nan, Kian M. Chai, Wee S. Lee, and Hai L. Chieu. 2012. Optimizing F-measure: A tale of two approaches. In *International Conference on Machine Learning*, John Langford and Joelle Pineau (Eds.). 289–296.
[21] Felix Naumann and Melanie Herschel. 2010. *An Introduction to Duplicate Detection*. Morgan & Claypool Publishers.
[22] Daniel Paull. 2003. *A Geocoded National Address File for Australia: The G-NAF What, Why, Who and When. PSMA Australia Limited, Griffith, ACT* (2003).
[23] Linda Williams Pickle, Lance A. Waller, and Andrew B. Lawson. 2005. Current practices in cancer spatial data analysis: A call for guidance. *Int. J. Health Geograph.* 4, 1 (13 Jan. 2005), 3.
[24] Robert Tibshirani, G . James, D. Witten, and T. Hastie. 2013. *An Introduction to Statistical Learning-with Applications in R*. Springer.
[25] William E. Winkler and Yves Thibaudeau. 1991. *An Application of the Fellegi-Sunter Model of Record Linkage to the 1990 US Decennial Census*. Technical report. US Bureau of the Census.

Received October 2017; revised April 2018; accepted June 2018