Data Preparation for Duplicate Detection

IOANNIS KOUMARELAS, LAN JIANG, and FELIX NAUMANN, Hasso Plattner Institute, University of Potsdam, 14482 Potsdam, Germany

Data errors represent a major issue in most application workflows. Before any important task can take place, a certain data quality has to be guaranteed by eliminating a number of different errors that may appear in data. Typically, most of these errors are fixed with data preparation methods, such as whitespace removal. However, the particular error of duplicate records, where multiple records refer to the same entity, is usually eliminated independently with specialized techniques. Our work is the first to bring these two areas together by applying data preparation operations under a systematic approach prior to performing duplicate detection.

Our process workflow can be summarized as follows: It begins with the user providing as input a sample of the gold standard, the actual dataset, and optionally some constraints to domain-specific data preparations, such as address normalization. The preparation selection operates in two consecutive phases. First, to vastly reduce the search space of ineffective data preparations, decisions are made based on the improvement or worsening of pair similarities. Second, using the remaining data preparations an iterative leave-one-out classification process removes preparations one by one and determines the redundant preparations based on the achieved area under the precision-recall curve (AUC-PR). Using this workflow, we manage to improve the results of duplicate detection up to 19% in AUC-PR.

CCS Concepts: • **Information systems** → **Data cleaning**; **Deduplication**; **Extraction, transformation and loading**; *Clustering and classification*;

Additional Key Words and Phrases: Data preparation, data wrangling, record linkage, duplicate detection, similarity measures

ACM Reference format:
Ioannis Koumarelas, Lan Jiang, and Felix Naumann. 2020. Data Preparation for Duplicate Detection. *J. Data and Information Quality* 12, 3, Article 15 (June 2020), 24 pages.
https://doi.org/10.1145/3377878

1 DATA PREPARATION

Difficult data cleaning tasks are typically treated with specialized algorithms. As an example, duplicated records that refer to the same entity with variations in their values represent a common error in datasets and are dealt with duplicate detection methods [12]. Depending on the application task, duplicate detection is referred to in the bibliography under different terms [9] including entity resolution, when it involves matching records within a single relation, or record linkage, when it involves matching records across two or more relations. For this work, we only focus on duplicate detection. To improve the performance of duplicate detection, researchers usually place

Authors' addresses: I. Koumarelas, L. Jiang, and F. Naumann, Hasso Plattner Institute, University of Potsdam, Prof.-Dr.-Helmert-Strasse 2-3, 14482 Potsdam, Germany; emails: {ioannis.koumarelas, lan.jiang, felix.naumann}@hpi.de.

1936-1955/2020/06-ART15 $15.00
https://doi.org/10.1145/3377878

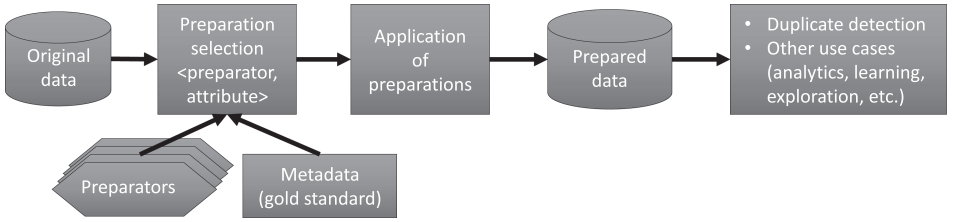

Fig. 1. Data preparation vision. A pool of preparators is available for many different operations that test and select the best of them w.r.t. the process at hand. The prepared data are then exported.

their effort in improving individual parts of the method itself. In contrast, we decided to regard a sometimes-overlooked aspect of duplicate detection, namely the preparation of the data before execution starts. We have found this latter issue to be prominent in the literature (refer to Section 2 for more details), where most systems are inconsistently preparing their input data, and thus it is impossible to perform any accurate comparison of their results.

In this article, we look at typical issues with datasets before deduplication and fix them by applying relevant data preparation operators (*data preparators*), in a systematic way. In particular, given a list of data preparation operations $P = \{P_1, P_2, \ldots, P_n\}$ and a dataset with attributes $A = \{A_1, A_2, \ldots, A_m\}$, our objective is to systematically find the list of the most beneficial combinations $<P_i, A_j>$ for duplicate detection among all possible solutions. We envision, and explain further in future work (Section 7), that this discussion will result in tools that, depending on the use cases, can select and apply the appropriate data preparators in an automatic way. This could happen either with prior knowledge or in an empirical way, as we do in this article, by applying and evaluating the improvement afterwards. The process could be similar to the one shown in Figure 1 and, depending on the end application, will need some extra metadata, such as a labeled set of duplicates and non-duplicates for our use case of duplicate detection. Such tools are currently not available, as most of the existing ones have a number of limitations (Section 3). First, they provide only a very limited set of data preparators, which are too generalized and typically not really helpful for specific difficult tasks, such as duplicate detection. Second, they require labeled sets for the data preparation itself, including pairs of unprepared and prepared values, which is an additional time consuming step.

A fundamental aspect of research is the ability to repeat and compare experimental results from different approaches. In this way, when an approach manages to perform better than the previous state of the art, we consider it a scientific achievement. However, in many cases experiments are performed on a different basis than the ones they are comparing themselves against. In particular, basing data cleaning and deduplication experiments on a consistent shared dataset with a ground truth has proven to be difficult. But also in many other areas of computer science, experimenters "massage" the data (usually in good faith) before applying their novel algorithms. We consider this to be a vital issue and have found many cases where this phenomenon takes place (see Section 2 for more details). Thus, not only are experimental results incomparable, but it might be that some approaches, simpler to understand and implement, may seem to underperform, whereas in fact they would perform better had they been given the same input.

A number of different steps are performed in the early stages of any data-driven application, often described as extract, transform, load (ETL) phase, including data preparation and duplicate detection [9, 24, 32]. Typical examples of data preparations are shown in Table 1, where character capitalization, special character removal, address normalization, and acronymization, transform the data into a more usable state. For more semantics-based issues, such as duplicate entries for a single entity or erroneous data, more sophisticated methods of *duplicate detection* and *data*

Table 1. Examples of Data Preparation

Preparator	Before	After
Capitalize characters	NEW york	NEW YORK
Remove special characters	<...> Berlin </...>	Berlin
Normalize address	V AVE, NY	Fifth Avenue, New York
Acronymize	United States	US

cleansing have to be applied. Merely syntactic data preparators are not enough. Our goal is complementary to these duplicate detection methods as we try to improve their effectiveness by a systematic preprocessing phase.

Typically, duplicate detection can be improved by adapting special parts of its process: (1) using better indexing frameworks to retrieve similar candidates that are not completely identical, (2) with similarity measures that can better quantify the similarity on each attribute for the previously retrieved candidates, and (3) by using better classifiers that given the previous similarities make satisfactory decisions. To focus on the effects of data preparation on duplicate detection, we decided to use only basic duplicate detection methods; blocking for indexing, the Monge-Elkan similarity measure, and a threshold-based classifier [9]. We discover those data preparators that convert the data to a more appropriate and clean state, before the actual process of deduplication takes place. In this way, we avoid the development of complicated methods afterwards, bloated with special cases. The overall procedure is not only more accurate but also faster, which is a prerequisite for many scenarios, such as those of real-time applications. Our basic deduplication configuration is comprised of blocking [4] for indexing (see Section 6.1.1), the hybrid similarity measure Monge-Elkan, using Levenshtein [30] as the token similarity measure (see Section 6.1.2), and a threshold-based classifier (see Section 6.1.3). We believe that similarly to deduplication, application of an initial data preparation can improve other processes as well.

Our approach. In a nutshell, we prepare the data for deduplication as follows (as explained in detail in Section 5): First, we sample the set of duplicate and non-duplicate pairs (Section 6.1.1 explains how we generate them). Second, we decide which combinations of preparators and attributes are valid that we call *preparations*, and apply them. Third, we keep only preparations that affect the similarities in a positive way, which indirectly suggests an improvement on classification. We call these *candidate* preparations. Fourth, the candidate preparations are reduced by applying leave-one-out optimization, leaving us with the *minimized* set of preparations. Fifth, we perform classification on the original and prepared data, and decide whether our process was helpful. Finally, the prepared data are exported so that the next phase of the complete duplicate detection and resolution can begin, which is not in the scope of our article.

The source code and the datasets used in the evaluation are available on our website.[1] Our contributions can be summarized as:

- A collection of data preparation examples in the literature.
- A systematic approach to find the most suitable preparations for a given deduplication task.
- Experiments to verify our findings from different perspectives.

Organization. We proceed with Section 2, in which we have collected examples of data preparation throughout many papers on duplicate detection. Section 3 covers relevant background for

[1] https://hpi.de/en/naumann/projects/repeatability/duplicate-detection/data-prep-for-duplicate-det.html.

the reader, in the areas of data quality, data preparation, and duplicate detection. Next, Section 4 introduces several real-world datasets, which were the inspiration for the selection of common data preparators presented threreafter. We then describe the core process of applying and selecting preparators in Section 5. Section 6 discusses the results of the process and its impact on duplicate detection. Section 7 concludes with our vision on the role of data preparation and how it could be incorporated in a variety of applications.

2 DATA PREPARATION IN THE LITERATURE

The general issue of massaging data before evaluation has been identified in the past. Here, we focus on the domain of duplicate detection, also known as record linkage, entity resolution, and so on [9]. First, the issue of incomparable evaluation results in publications has been identified by Köpcke et al. [22], who point to publications that use different sizes of the Cora dataset, which we discuss in Section 4.1. Second, Vatsalan et al. recognize the importance of data preparation steps, prior to performing qualitative record linkage, such as enrichment, removal, and transformation of records' values [36]. Finally, the inability to reproduce experiments of a given work is a major issue in the data cleansing field. One is that many authors do not reference the used datasets, or refer to them with ambiguous names. Another aspect is that sometimes private datasets are used, which are not made publicly available with appropriate documentation. Indeed, we also consider such a private dataset, which we cannot disclose, but believe that the findings from it are sufficiently interesting. Finally, it is not always clear how the training and testing datasets are used, and whether parts of the latter are used for the training phase as well.

To motivate the importance of data preparation in duplicate detection, we have selected quotes from papers in the general area of duplicate detection, in which the authors perform some form of preparation steps on the data. In particular, we also included literature where these preparation steps are not clearly specified, which makes reproducibility particularly difficult. While we focus on duplicate detection, massaging data is a common practice for all of data science. The two most common preparation steps done on the datasets are (1) selecting a subset of the dataset and (2) altering the content, through the use of some preparator. Our focus is on the latter type. We begin with cases of well-explained data preparation steps. The extent of this selection is also intended to show the wide variety of data preparation steps.

> Reference [26]: "...we thus increased the hardness by deriving a data set with only first name and last name initial for each inventor. We call the original data set the full set and the derived one the partial set...."
>
> Reference [37]: "We concatenated attribute values of each citations to a string..."
>
> Reference[27]: "A set is a publication; tokens are character q-grams of the concatenated title and author strings (q = 2, case insensitive)...."
>
> Reference [23]: "In order to obtain the full feature set, we join the comments and authors CSV input files. To create a vector representation, we apply feature hashing to the authors name and the comments text. We split the name by camel case, white space, and other delimiters and hash the words to a fixed size feature space. For the comments, we split the text into words and hash them as described before."
>
> Reference [35]: "We process the collection by removing pages that have size less than 5 KB, contain no URLs, have an exact duplicate (identified using a standard hash function), or consist of non-English words, all of which shrinks D to a total of 70M pages. We parse each remaining page, removing stop-words and stemming all text outside of HTML tags. We then create feature vectors with weights $t_i(u)$ being the normalize TF-IDF score of each word i on page u..."

Reference [3]: "When preparing the data, the attribute values were first normalized using general substitution rules (for example, uppercase to lowercase conversion), and then segmented into tokens, either characters or words, depending on the attribute types and the description languages (for example, names written in Japanese characters were segmented into characters, and the titles were segmented into words)…"
Reference [10]: "This paper describes an implementation of lexicon-based tokenisation with hidden Markov models for name and address standardisation…"
Reference [10]: "Training of HMMs for residential address standardisation was performed by a process of iterative refinement…"

While the mentioned publications do explain their preparation steps, they remain ambiguous in some cases. The following are selected examples of more unclear preparation procedures:

Reference [39]: "Prior to actual duplicate detection, we performed data standardization, which consisted in removing special characters from string attributes."
Reference [6]: "For the company names dataset, we also inject domain specific abbreviation errors, e.g., replacing Inc. with Incorporated and vice versa."
Reference [10]: "Name standardisation. To assess the accuracy of name standardisation, a subset of 10,000 records with non-empty name components was selected…."

We split the literature w.r.t. the level of explanation given. Most probably though, it is expected that people's opinions differ on what is a good explanation or not. In any case, we believe that there is some space for improvement, and in particular for a systematic approach that makes the process of deciding data preparation steps easier to explain and justify.

3 RELATED WORK

There are many and various approaches to resolve *data quality* issues that exist in real-world data. These attempts include, among others, data transformation, in case of different file formats, encodings, attribute names and types, data standardization (or normalization), so that the values conform to the uniformly agreed rules, and data integration, to merge different sources of data. *Data preparation* is a general term we use to refer to all these different tasks, as most of them are not always easily distinguishable.

Data quality. To begin with, data quality has been broadly studied over the years [15, 24, 38]. Since it represents a core part of the business intelligence (BI) model, there has also been work that explains steps from that perspective [34]. A taxonomy of the different quality issues that appear in data has been proposed in the past [31, 33], with the number of different integrated sources being one leading cause of errors.

Data preparation. A clever selection and application of *data preparators* on data can transform the latter to a cleaner state. For this reason, data preparators are an essential part of this article. the authors of Reference [21] under the term of data wrangling consider different actions to *transform* data, *manage* these transformations, and even *scale* them in the cloud. A large part of this work also considers interactive visualization, which they consider to be a vital part of this process.

Yang et al. introduce the extract transform load (ETL) framework Lens and the concept of lenses, which are data processing components [40]. An example lens is domain constraint repair, where constraints, such as "NOT NULL," are applied to prepare the data.

In cases where user input is possible by providing input/output examples of unprepared/prepared alphanumeric values, the authors of References [14, 20] have proposed methods for synthesizing functions, which learn how to perform these transformations and apply them to the

remaining data. Both approaches solve the issue of searching through a large space of possible solutions by incorporating heuristics and pruning rules, which form functions comprised of string operations. These string operations, examples of which are *substring*, *concatenate*, and *split*, extract and then transform the input to the desired output format. In the same spirit, with the user providing input/output examples, the work in Reference [17] utilizes over 50K functions that transform an input value into the desired format from GitHub and Stackoverflow along with examples from Wikipedia tables to synthesize its programs. Using a number of representative examples for each data type, such as dates and names, in combination with the extracted functions, it finds those functions that fulfill the criteria of the provided input-output examples, and finally synthesizes them into a program that performs this transformation. For instance, consider the task of date normalization, given a set of input-output examples, where all output examples are dates following the U.S. format, while the input examples follow a variety of date formats, it finds a function that parses the input date and transforms them into the U.S. format. Finally, DataXFormer is similarly based on user-provided input/output examples, but expects as input a relational table with the goal of transforming it into a format denoted by the examples [2]. This is done by utilizing Web Tables[2] to find the most similar tables and rows, which share a number of attributes with the input table. Thus, for every row of the input table, a number of rows from Web Tables provide attribute values of better quality.

Before any repair can be performed, the errors have to first be identified. The authors of Reference [1] experiment with a number of different tools, such as OpenRefine[3] and Trifacta[4] on the basis of recognizing specific types of errors on a number of different datasets.

Duplicate detection. Research on identifying and resolving duplicates has been present for decades, as the problem appears in many applications. This has led to some commonly applied steps [9, 12, 18, 30] that are used to identify such entries, which are to (1) prepare the data, (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. In this article, we are focusing on the first part, *data preparation*, suggesting that the latter steps are easier and more successful after data preparation. For the rest of the steps ((2)–(5)), we just use standard deduplication techniques (see Section 6.1). Most of the current literature focuses on the latter steps with different techniques, having accepted the fact that input data is not in good shape.

First, during the *preparation* step, typically applied techniques [9, 12] include data standardization and enrichment through external data sources, removal of unwanted characters, and segmentation of compound values into separate attributes. Second, regarding the second step, *indexing*, Blocking and Sorted Neighborhood [9] represent two of the most successful methods for retrieving candidates. Third, typically for the *comparison* of the retrieved records, similarity measures are used. For instance, for alphanumeric attributes Levenshtein, Jaccard, and Monge-Elkan [9] are commonly applied as they represent three different categories of edit-based, token-based, and hybrid similarity measures. More recent works focus on machine-learnable similarity measures, where by using custom features or letting artificial neural networks learn the features themselves, and then further tailor the similarity measures to the data [7, 29].

Fourth, for *classification*, the threshold-based classifier is commonly used, as its usage is transparent and simple. Given a pair of records, and a list of its calculated attribute' similarities, it linearly combines them and if the sum is above a specified threshold, then the comparison represents a match, otherwise not. Searching the parameter space on this classifier is equal to finding the

[2] http://webdatacommons.org/webtables/.
[3] https://github.com/OpenRefine/OpenRefine.
[4] https://www.trifacta.com/start-wrangling.

threshold and optionally weights for each attribute's similarity (if not all attributes have the same impact). Other classifiers have also been used, such as decision trees [11] and support vector machines (SVM) [7]. Finally, for *evaluation*, a classic calculation of true positives (TP), false positives (FP), and false negatives (FN) is enough to calculate precision, recall, and F-measure. In addition, area under the precision-recall curve (AUC-PR) has been proposed as an alternative to assess detection quality. The F-measure does not fairly balance the errors of the two classes, duplicates and non-duplicates [16]. In contrast, the AUC-PR, considers all possible thresholds and provides a more holistic evaluation. Using the precision-recall curve it is further possible to calculate the best combination of precision-recall and thus F-measure.

However, in reality any problems that are not fixed during the data preparation phase can and must be addressed in later steps. For instance, a complex similarity measure can address inconsistent formatting of phone numbers. With careful preparation, later steps are more easy to implement and are more effective. Therefore, our article focuses on identifying the appropriate preparations that need to be applied, to increase the effectiveness of the following phases, regardless of which methods are used.

4 DATA QUALITY AND DATA PREPARATION

How to identify and repair specific issues in datasets is the focus of this section. First, we introduce a number of datasets that we used to find and resolve issues. This helps our discussion: Whenever possible, we mention real examples from these datasets. Second, we propose preparators suitable for deduplication, along with some short description and examples.

4.1 Datasets

Identifying issues existing in real-world problems demands the usage of real datasets. For this reason, we introduce such datasets with a brief description, and discuss the attributes we used for similarity calculation (see Section 6.1.2) and consequently as features for pair classification. The attributes' selection was based on completeness (percentage of records with a non-null value) and usefulness, which means we omit attributes we consider redundant or not relevant for data preparation. Some further information w.r.t. the gold standard that is needed for the experiments with the generation of non-duplicate (NDPL) pairs, is discussed in Section 6.1.1. Details about all the datasets, as well as download information for four of them are available in our website.[5] Last, statistics about the number of records, duplicate (DPL) and non-duplicate (NDPL) pairs can be found in Table 2.

- CDDB contains entries of audio CDs with descriptive attributes, such as artist, title, tracks, genre, and year. From the eight available attributes, we use artist, category, genre, title, tracks, and year. Tracks is a compound element, in which individual track values are concatenated using " | " as the separator.
- Census is based on real data, generated by the U.S. Census Bureau, and contains a single attribute with the record's value, called text. This dataset contains two relations A and B, and thus could be also used for record linkage, i.e., linking the two relations. However, for our purposes we treat it as a typical single-relation dataset and try to find the duplicates instead of the linkage matches. The single attribute text includes information about last_name, first_name, middle_name, zip_code, and address.
- Cora is comprised of bibliographic records about machine learning publications and includes relevant information in 17 attributes, from which we use authors, journal, pages, title, and year.

[5] https://hpi.de/en/naumann/projects/repeatability/duplicate-detection/data-prep-for-duplicate-det.html.

Table 2. Datasets Statistics on Number of Records, Duplicate (DPL), and Non-duplicate (NDPL) Pairs

Dataset	Records	# DPL	# NDPL
CDDB	9,763	300	8,944
Census	841	376	1,631
Cora	1,879	64,578	176,285
Hotels	364,965	94,677	368,002
Movies	39,180	14,190	14,069
Restaurants	864	112	11,460

NDPL pairs are generated according to the blocking process described in Section 6.1.1.

- Hotels is provided by our industry partner, which contains information about hotels across the globe. We limited our experiments from twenty-seven attributes to hotel_name, street_address, zip_code, city, state, and country.
- Movies is the result of merging two different datasets, based on IMDB.org and film-dienst.de, such that real-world duplicates are available. The information provided is limited to the attributes of actors and movie title, which we both use.
- Restaurants is a merged dataset of two other relations, based on Fodor's and Zagat's restaurant guides. From the six available attributes, we use name, addr, city, phone, and type.

Cora contains large clusters of DPLs, which means many variations of entities, and Hotels, which is a real-world industry dataset, contains more difficult variations of DPLs, as hotel names and their addresses are provided in many different formats, with a lot of ambiguity. For instance, in hotel names sometimes only the chain is provided, whilst addresses are not always fully specified, with records missing important information, such as the city. CDDB, Census and Hotels gold standards of DPLs were extended with some further pairs, obtained by transitive closure and exact match on the entire record.

4.2 Preparators

A preparator is a method that transforms a set of input values into a set of output values that are of higher quality or more useful for the use-case at hand. A preparator's complexity can vary from being fairly simple, such as upper-casing all strings, to being quite complex, such as geocoding address fields. The number of input and output attributes can vary and be of any datatype, although in this work we focus only on alphanumeric values.

Based on this definition, in this section we present a number of preparators that we consider to be suitable for duplicate detection and have implemented. The presented preparators are the ones we deemed meaningful for the sample of datasets we used and do not represent a list of all possible preparators for duplicate detection. Brief descriptions of the implemented preparators along with short examples are shown in Table 3. As we denote in Table 3, the first six preparators are *lossless*, whereas the last five are *lossy*. Lossy preparators are particularly useful for data matching tasks, such as duplicate detection. Further information along with the description of their slightly different application is provided in Section 5. We proceed by providing an explanation of them and whenever possible accompany them with examples from the datasets presented in the previous section.

Table 3. Data Preparation Operators, Sorted by Their Intended Order of Application

Preparator	Description	Example
Split attribute	Extract parts of an attribute, moving them into other attributes	`10117 Berlin, Germany` → ZIP-code: `10117`, City: `Berlin`, Country-name: `Germany`
Normalize address	Convert address to its commonly accepted form, fixing inconsistencies	`fredrich stret 43, berlin, German` → `Friedrichstrasse 43, 10117 Berlin, Germany`
Geocode	Get the geolocation of an address	`Friedrichstrasse 43, 10117 Berlin, Germany` → latitude: 52.5071081, longitude: 13.3896519
Remove special characters	Remove non-alphanumeric characters: [,.:;"^'/\!@#$%&*()_+=|<>?{}[]~-]	`'/ > @ Berlin! <'` → `Berlin`
Transliterate: UTF-8 to ASCII	Remove diacritics from words	`München` → `Munchen`
Merge attributes	Merge multiple attributes into a single one	(opposite of split attributes)
Acronymize	Keep the first character of all tokens	`Very Large Data Base` → `VLDB`
Capitalize characters	Convert all characters to upper case	`BERlin` → `BERLIN`
Syllabify	Word → syllables	`preparation` → [`prep`, `a`, `ra`, `tion`]
Phonetic encode	Convert value to its pronunciation representation	`Berlin` → `BRLN`
Stem	Reduce word to base form	`programming` → `program`

The first six preparators have a lossless effect on the applied values, whereas the last five are lossy. The latter ones are particularly useful in duplicate detection and when applied sacrifice a large portion of the initial values, in a way that is necessary to reveal duplicate values.

4.2.1 Split Attribute. While for some applications a coarser schema is appropriate, other applications need more fine grained values. For instance, to prepare data, a name field might need to be split into first name and last name. The correct solution is to identify such cases, extract and move these data to their respective attributes. Census represents a good example of this category and is used in our experiments as it contains a single attribute (text) with a mixture of information. To allow for a more fine-grained comparison, we decomposed the attribute text into new attributes of last_name, first_name, middle_name, zip_code, and address. This is achieved by using metadata that specify the ranges of substrings required to extract these values. For instance, last_name is contained within the range of characters [0, 15), first_name follows in the range of [15, 28), and so forth. The retrieved values are trimmed, to remove excess whitespaces before and after the values. Please note that there can be more complicated situations, where the values are not easily separable, the attributes' order is not the same, or the phenomenon is only present on a subset of records.

4.2.2 Geocode. Geo-coordinates (latitude, longitude) are useful for the deduplication process for data with geographic information. A pair of records whose distance of their geocodes is greater than some threshold can be pruned, because they are too far away to represent the same entity. Such a threshold can range from a few meters to some kilometers, depending on the application.

The process involves the usage of Geographic Information Systems and the operation of *geocoding*, which, given the address information, returns the geo-coordinates of the location. For our experiments, we used the Nominatim system.[6] In the same context another available preparator is *address normalization*, which given a noisy address retrieves the correct, standardized value. We considered *geocoding* and *address normalization* on Hotels, which includes detailed addresses improperly formatted. By focusing on street addresses, address normalization retrieves values' extended versions, for instance "585 PKWY Dr NE" normalizes to "585 parkway drive northeast" and "2361 W NW Hwy" to "2361 west northwest highway."

4.2.3 Remove Special Characters. Bad parsing or conversion can propagate issues to next steps. Web scraping and extracting values with optical character recognition (OCR) are examples where irrelevant and redundant characters may be erroneously identified as part of the attribute's value. In such cases removing them leaves us with the pure intended value. For instance, in Cora we observe several instances where redundant parentheses, full-stops, and hyphens are left-overs from parsing. For instance, in the attribute Year we meet values, such as "(1987)," pages with values similar to "(pp. 24–30)," and publisher with cases equivalent to "American Association for Arti-ficial Intelligence," where the hyphen probably meant a continuation to the following line.

4.2.4 Transliterate (UTF-8 to ASCII). Transliteration is the process of converting a text from one format into another. Its purpose is to preserve the characters to the most similar ones in the destination language, which in our experiments is English. For our application, we aim at removing completely any diacritics, which convey a special meaning, but make the deduplication task harder. Examples here include a conversion of München to Munchen and Café to Cafe.

4.2.5 Merge Attributes. During the data entry phase, errors due to improper placement of values across the attributes might be introduced. In those cases, we merge all the participating attributes into a single one using a whitespace character in between the merged values, effectively treating it as a single long string, with the goal of alleviating a number of errors during duplicate detection. Alternatively, we could instead merge a subset of attributes into a single new one. For instance, in Hotels we could merge the attributes that describe the address into a new single attribute address.

4.2.6 Acronymize. An *acronym* is a type of abbreviation or else shorter form of a phrase, which is typically performed by keeping the first character of all tokens. For instance, `Data Preparation` → `DP`. By applying this preparator, we aim at identifying those cases where people were not consistent in submitting the full version of the word or the acronymized one. In contrast, *abbreviations* are a superset and can include cases where the short form is selected with a different reasoning, such as first and last characters, phonetic encoding, or something else. An example abbreviation is `Street` → `Str.` Abbreviations are domain specific and thus a careful consideration has to be taken when expanding an abbreviated value into its full form. This preparator is useful in cases where some people might know a place with its short form or consider it too large for a field. Given the multitude domains of our datasets' attributes, abbreviations are currently not supported.

4.2.7 Capitalize Characters. Different *capitalization styles*[7] are used, mainly the *sentence case*, *title case*, and *full capitalization* (all-caps), to denote a specific meaning. To overcome inconsistencies, such as data entry errors, we choose to use the latter by upper-casing, and lose information whilst increasing the similarity. For instance, in research titles of Cora one might come across

[6] https://nominatim.openstreetmap.org.
[7] https://en.wikipedia.org/wiki/Capitalization#Names_of_capitalization_styles.

versions, such as "Small disjuncts in action: Learning to diagnose errors in the telephone network local loop" and "Small Disjuncts in Action: Learning to Diagnose Errors in the Local Loop of the Telephone Network."

4.2.8 Syllabify. Segmentation decomposes a compound word into the simpler ones, which is a case that can happen when the user is not completely sure of the proper form. Such examples include, `database` → `data base`, `dataset` → `data set`, and `football` → `foot ball`. A more specific type of segmentation, which we consider in our experiments, is *syllabification*, where a word is split into its syllables, usually for visual reasons, such as in fields that depend on resizeable user interfaces. In such interfaces, this introduces a continuation of words in the following lines, particularly useful in larger values, such as the address, which in some countries can be truly long.[8] As an example, `Friedrichstrasse 43, 10117 Berlin, Germany` → `Fried rich stras se 43, 10117, Ber lin, Ger many`. For our experiments, we used English syllabification patterns. Alternatively, if support of more languages is desired, a combination of language detection with specialized patterns for each language is possible. Multilingual approaches are beyond the scope of this work.

4.2.9 Phonetic Encode. Phonetic encodings aim at converting words into a representation that mimics their pronunciation. Different encodings have been implemented throughout the years, including Soundex and Metaphone [9], which we use in our experiments, and represent a successful step toward deduplication. For instance, in CDDB, using Metaphone for "Metallica" we obtain "MTLK," which is also the same for "meettaaliica." Strictly speaking, such lossy preparations are very specific to the problem of duplicate detection, where they perform well, as they are able to abstract from the original values.

4.2.10 Stem. Stemming helps overcome different word endings, which is often the case when attempting to transcribe information through phone or from a speech. Being a simple model it attempts to retain only the base of a word, without incorporating any contextual information. Examples here include "discussing" and "discussion" that are stemmed to "discuss." For our experiments, we used the English *Porter stemmer*.[9] Similarly to SYLLABIFY, multilingual support is possible, but is beyond the scope of this work.

In this section, we described preparators that we considered to be the most impactful and that is why we implemented them for our system and experiments (see Sections 5 and 6). However, there are many more possibilities for other preparators, even in duplicate detection, which we did not implement, such as:

- Normalize person names, phone numbers, and dates: parsing and formatting values according to a specified format. For instance phone numbers could be represented in E.164,[10] which is a worldwide standard that can universally represent phone numbers.
- Encode characters to UTF-8: Converting different character encodings to UTF-8 would allow for proper value storage and comparison.
- Generate character n-grams: having a list of n-grams from a given alphanumeric value could assist a fuzzy indexing, where similar, but not necessarily the same, values could be retrieved.
- Standardize NULL value representation: replace values that represent lack of a value, such as "n/a", ".", "-", and "not provided", with the empty string "" or some other predefined value.

[8] https://en.wikipedia.org/wiki/List_of_long_place_names.
[9] https://tartarus.org/martin/PorterStemmer/index.html.
[10] https://en.wikipedia.org/wiki/E.164.

- Remove stop words: such as “a”, “the”, “to”, as these increase the similarity unnecessarily in non-duplicates.
- Convert categorical or ordinal to numerical: converting values to numerical using a predefined mapping or ordering. For instance, genders could be mapped to $\{0, 1\}$, whereas days of week could be represented with a mapping to $[0, 6]$.
- Normalize numerical values to [0, 1]: a normalization using the minimum and maximum values.
- Apply transformation rules: replacing or removing specific words. For instance, in the case of synonyms replacing values with one representative could help improve similarities among duplicates. Alternatively, values that have been erroneously entered can be completely removed.

5 DEDUPLICATION PREPARATION

In this section, we describe our process for trying a number of available preparators on a dataset’s attributes and recommending the pairs of preparators and attributes that lead to improved results. We discover the most important preparations using a heuristic 2-step pruning process: based on a (small) gold standard. A heuristic approach is required, because examining all possible preparation combinations is infeasible. Thus, our process cannot guarantee an optimal result set. Based on these assumptions our process works as follows: First, we apply a fast process based only on similarities, which reduces the search space drastically. Then, we follow with a slower process that bases its decisions on whether to keep preparations using duplicate detection classification on the prepared values, which is our final goal. An exhaustive attempt, trying all preparations could be possible, but is not feasible in real scenarios with large datasets. Our process is shown in Figure 2 and can be decomposed in the following phases:

Input: The user provides data, optionally some constraints to specify which preparators are not valid for certain attributes, and a set of labeled pairs (gold standard), whereas the system provides its available preparators. In case a constraint was not provided for a preparation that is meaningless, such as ADDRESS NORMALIZE on a first_name the system would produce unhelpful results. By selecting the 10% most difficult pairs, we create a silver standard. This translates to the 10% duplicates with the lowest similarity and 10% of non-duplicates with the highest similarity. This helps us represent a realistic scenario, as in practice before a larger and complete set of pairs (gold standard) can be collected, users typically start experimenting with smaller sets (silver standard) that include difficult cases. In domains where labeling is more expensive, users can further reduce this sampling ratio, assuming they still make sure to include challenging pairs. Moreover, since almost all of the process’ steps are performed on sampled data, the execution time stays within reasonable limits.

Phase 1 – Baseline: In this phase, we are interested in performing classification on the original data, so that we can later judge if the application of preparations helped or not. This classification is performed in two steps. First, we calculate the pair similarities for the silver standard, based on the data. Second, we perform classification using the pair similarities as features, evaluate the results, and determine the *AUC-PR*.

Phase 2 – Preparation application: Given the set of preparators $P = \{P_1, P_2, \ldots, P_n\}$ and attributes $A = \{A_1, A_2, \ldots, A_m\}$ this step produces all valid combinations $<P_i, A_j>$, called *preparations*. We call a combination valid if the data type of the attribute is applicable to the data type that the preparator is expecting. For instance, it is pointless to apply “geocode” on a first name. This information is provided as metadata at the input phase. Finally, for each preparation we calculate the pair similarities.

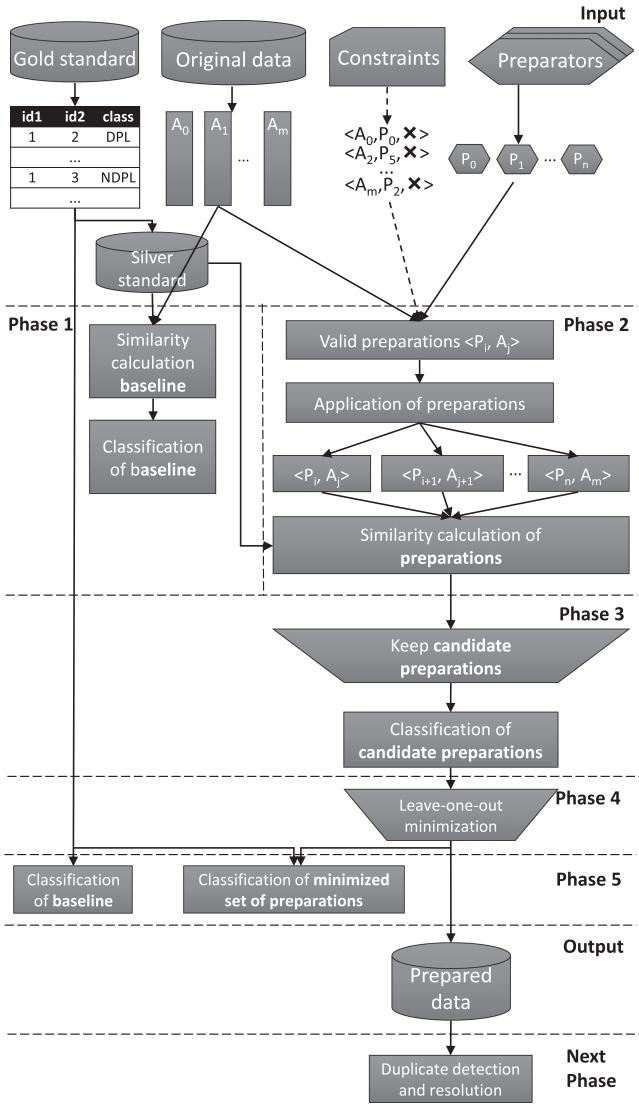

Fig. 2. Process of identifying useful preparations for duplicate detection.

Phase 3 – Candidate preparations: In this step, we aim to filter preparations w.r.t. their effect on similarities. This filtering approach is intuitive, since classifiers use the similarities as their features. Given the N duplicate pairs and M non-duplicate pairs of the silver standard, we keep preparations that fulfill the following condition:

$$\frac{\sum_{i=1}^{N} \delta sim(\text{DPL[i]})}{|\text{DPL}|} - \frac{|\text{NDPL}|}{|\text{DPL}|} \cdot \frac{\sum_{i=1}^{M} \delta sim(\text{NDPL[i]})}{|\text{NDPL}|} > 0$$

$$\iff \sum_{i=1}^{N} \delta sim(\text{DPL[i]}) - \sum_{i=1}^{M} \delta sim(\text{NDPL[i]}) > 0,$$

where δsim expresses the similarity delta of a pair before and after preparation ($sim_{after} - sim_{before}$) and becomes negative when the similarity drops. Regarding the second formula, we calculate the sum of these similarity deltas, for the duplicate (DPL) and non-duplicate (NDPL) pair set, respectively. The intra-similarities of entities, which are described by DPL pairs of records have to increase, and in contrast inter-similarities that are in turn described by NDPL pairs of records have to be decreased, or at least increase less than the DPL ones. After this filtering, we are left with the *candidate* preparations, with which we proceed to perform another classification.

This phase evaluates and decides which individual preparations are interesting to be considered as candidates. The order of application is an additional dimension that affects the results of the following two phases (phases 4 and 5). We do not address this problem here but leave it for future work. Instead, we proceed by applying all preparations using a *predefined* ordering, namely, as they appear in Table 3. As an extension of the order issue, some preparators are helpful for our domain specifically, but lossy in the sense that important information is removed. We do not consider this term in an information-theoretic sense, but rather in a data quality sense: values prepared with lossy preparators are typically not useful in use-cases other than matching-related ones. These lossy preparators are the last five of Table 3. We execute them after the others, creating new columns with the transformed values. For instance, if the preparators NORMALIZE ADDRESS, TRANSLITERATE, and ACRONYMIZE are selected for the attribute city, we first prepare city using NORMALIZE ADDRESS and TRANSLITERATE, creating a respective column. ACRONYMIZE, which is a lossy preparator, is applied on top of the former column, creating an additional column with only the initials. By not considering lossy preparators in combination, we risk missing some corner cases where a combination of two such preparators would help reveal duplicates. In the majority of cases, however, applying one lossy preparator after another changes the value so drastically that it is likely useless for our task, introducing more matching mistakes.

Phase 4 – Leave one out optimization: Some preparations might have no effect after the application of other preparations. For instance, by considering the examples of Table 3, applying REMOVE SPECIAL CHARACTERS on '`/> @ Berlin! <`' returns the value `Berlin`. We would observe the same effect on the value if we had instead applied NORMALIZE ADDRESS. Thus applying both preparations is redundant. On the same principle, applying some preparations in combination can also worsen the result. Understanding the effect of a preparator on a pair of records is more complicated, as it is a combination of the pair's class (duplicate or non-duplicate), and its initial and final similarities. Thus, to make our final decisions and remove such preparations, we perform an optimization process. In this optimization process, we remove preparations to avoid redundant preparations (similarity stays the same) or preparations that are in fact harmful (similarity increases after removing them). We iteratively leave out individual preparations and perform classification using the rest. For instance, given P candidate preparations, there will be P executions, where in each execution one preparation is removed, whilst the rest $P - 1$ preparations are present. This process could be also characterized as "backwards elimination with replacement." Removals that keep the AUC-PR constant or even improve it are made permanent, which leaves us with the *minimized* set of preparations.

Phase 5 – Final classification and comparison: After the second pruning of the previous phase, where we decided the minimized set of preparations, we re-perform classification, but this time on the full set of pairs (gold standard), to acquire our final metrics. Finally, we compare these metrics to the baseline (also on gold standard).

Output: By comparing the AUC-PR results from our process and the baseline, we can determine whether our process was helpful. If this is the case, then we proceed by using the prepared data from Phase 4. Otherwise, we can ignore our whole process and proceed with the original data.

Next phase: This is the main process that we performed the data preparation for. In our case, this should be equivalent to performing duplicate detection with a larger set of goals. For instance, using a larger set of pairs than the one provided in our process or resolving the duplications found by merging the duplicated records into a single one per entity. Resolving all issues around duplicate detection is not in the scope of this article.

6 EVALUATION

We now examine the effect that different *preparators* have across *datasets* from two different perspectives, first by focusing on the preparators and second on the datasets. Examples of these preparations across datasets are shown in Table 4. We first explain some preliminaries in Section 6.1 to set the ground for the experiments described in Section 6.2. The findings are then summarized in Section 7.

6.1 Preliminaries

In the following sections, we discuss critical decisions. Briefly, in Section 6.1.1 we discuss the need for careful non-duplicate pair generation. Subsequently, Sections 6.1.2 and 6.1.3 discuss the choices we made regarding similarity measures and classifiers, respectively.

6.1.1 Gold Standard and Non-duplicates Generation. The gold standards that are available for many of the datasets in Section 4.1 each contain a list of record pairs that uniquely identify *duplicate record pairs*. In case this set is not transitively closed, we additionally create all transitive pairs and call this extended set *DPL*. To train and test a classifier, we also require a number of negative examples, i.e., non-duplicate pairs. When creating such non-duplicate pairs, we should make sure that records are not paired up randomly, since such pairs are expected to be very different and therefore trivial to classify. More useful for training and more realistic for testing, however, is to expose the model to pairs that are harder to classify, corresponding to a more realistic scenario. In practice, to speed up duplicate detection, blocking methods [9, 12] are typically used to divide datasets into disjoint subsets, called blocks, according to a predefined partitioning key. To avoid false negatives, usually multiple partition keys are defined. It is important to select the partition key with great care, as it controls not only the size and number of blocks created, but also how similar the individual data records within each block are.

We perform a simple blocking by using each attribute of the currently processed dataset as a partitioning key. Record linkage datasets are treated as deduplication datasets, i.e., pairs of the same source relation can be positioned in the same block. In case an attribute is multi-valued, we first divide it into its individual values and use those for blocking. Very unique attributes, which would lead to many single record blocks, are split into word or character n-grams of size 2 to 10 characters, depending on the length of the attribute, and then used as partition keys. The complete blocking scheme is discussed in Appendix A. As a result, we obtain several blocks, within which we create the cross product of all contained data records and thus form data record pairs, which are then combined into a common dataset. After making sure that there are no known duplicate pairs in the resulting dataset, we refer to it as *NDPL*. The NDPL dataset is, therefore, the set of all record pairs that can be formed within all blocks and does not include duplicate pairs.

6.1.2 Similarity Measures. Choosing similarity measures can be a tricky task. In the ideal scenario, if we can identify the type of an attribute, then we should select a similarity measure tailored for it. For instance, for an attribute that contains information about calendar dates, one could select a similarity that considers a difference in year a lot more important than a difference in days. To avoid complications, we use alphanumeric similarity measures, with the exception of a geolocation distance. For the latter, we use the Haversine [19] distance, and instead of using a

Table 4. Preparation Examples, Across Datasets

Dataset	Preparator	Attribute	Class	Before #1	Before #2	Sim Before	After #1	After #2	Sim After
cddb	phonetic encode	artist	DPL	VĂ¸mmĂ¸l Spellmannslag	V mm l Spellemannslag	0.46	FMLSPL	FMLSPL	1.00
			NDPL	The Beta Band	Raoul And The Big Time	0.53	0BTBNT	RLNT0B	0.00
cora	remove special characters	authors	DPL	Aha, D., Kibler, D., & Albert, M	D. Aha, D. Kibler, and M. Albert.	0.67	Aha D Kibler D Albert M	D Aha D Kibler and M Albert	0.86
			NDPL	Fahlman, S. E.,	Clouse, J., & Utgoff, P.	0.35	Fahlman S E	Clouse J Utgoff P	0.00
restaurants	syllabify	address	DPL	804 northpoint	804 north point st.	0.63	804 north point	804 north point st.	0.89
			NDPL	5937 geary blvd.	14928 ventura blvd.	0.50	5937 geary blvd.	14928 ven tu ra blvd.	0.39
movies	remove special characters	title	DPL	L'Intrus	Intrus, L'	0.52	LIntrus	Intrus L	0.65
			NDPL	Oh... Rosalinda!!	H.M.S. Defiant	0.26	Oh Rosalinda	HMS Defiant	0.06
hotels[1]	normalize address	city	DPL	CHICAGO		0.00	CHICAGO	CHICAGO	1.00
			NDPL	WILMINGTON		0.00	WILMINGTON	HERNDON	**0.30**
movies	acronymize	title	DPL	The Clinton Chronicles	Clinton Chronicles, The	0.97	TCC	CCT	**0.33**
			NDPL	Daredevils of the West	Desperadoes of the West	0.82	DOTW	DOTW	**1.00**
hotels	acronymize	hotel_name	DPL	HOTEL ADONIS	ADONIS HOTEL STRASBOURG	0.86	HA	AHS	**0.33**
			NDPL	INFUSION COFFEE SHOP	IP CASINO RESORT SPA	0.23	ICS	ICRS	**0.75**

Before and After represent the values before and after the application of the preparation, respectively (with #1 and #2 representing the two records' values). Examining the similarities ("Sim"), using the Monge-Elkan similarity measure (explained in Section 6.1.2), we show selected successfully and **unsuccessfully** (marked with **bold** characters) prepared pairs; duplicates (DPL) and non-duplicates (NDPL).

[1]ADDRESS NORMALIZE is using the rest address attributes of each record to decide what is the correct normalized address and thus city.

threshold to prune pairs with a larger distance we convert it into a similarity by using the formula $\text{similarity} = \frac{1}{\text{distance}^2+1}$, where distance is in meters. Using this formula smaller distances result in higher similarities. Therefore, pairs with large distances have a smaller similarity and less chances to be considered as duplicates, assuming always a successful prior application of GEOCODE.

In alphanumeric similarity measures, there are three main categories: edit-based, token-based, and hybrid. Examples of these categories are Levenshtein, Jaccard, and Monge-Elkan [30]. We selected Monge-Elkan for our experiments. Monge-Elkan goes through all tokens of the first input string and finds the token of the second string with the maximum similarity, using Levenshtein in our case, which can also be re-used for further matches [28]. Finally, it returns the mean similarity of the matched token pairs. Other categories of string similarity measures include phonetic similarity measures, such as SOUNDEX [9], and feature-based ones, such as MinHashing and Locality Sensitive Hashing (LSH) [25], which offer an approximation of the Jaccard similarity coefficient [9].

6.1.3 Classifier. We use a linear or threshold-based classifier (THR) [13], because it is a simple and effective model, that is frequently used in duplicate detection [5, 8]. The features we provide to the classifier are the similarities calculated for the attributes described in Section 4.1, using the similarity measures of Section 6.1.2. To keep our process simple, we choose the weights of the features to be equal. THR calculates a linear combination of the features and if the sum is above a threshold the classified label is 1 (match), otherwise 0 (no match).

Instead of selecting a single threshold to calculate the F-measure, which as discussed in Section 3 does not fairly balance the errors between duplicates and non-duplicates, we use area under precision-recall curve (AUC-PR). AUC-PR considers all possible thresholds, relevant to the evaluated pair similarities. Finally, since we do not choose any thresholds we use the full set of DPL and NDPL pairs for testing.

6.2 Effect on Similarity

In this section, we examine how preparators affect the similarity of values in various attributes as described in Phase 2 of Section 5. Before we analyze the collective results across datasets, we motivate the discussion by presenting a few interesting cases of preparations for exemplary values across five of the datasets. In Table 4, we show selected value pairs from different datasets and their similarity for a true duplicate and for a true non-duplicate, each before and after preparation. In the case of hotels, ADDRESS NORMALIZE is using the rest address attributes of each record to decide what is the correct normalized address. Using the normalized address, attributes such as the city, can be corrected or filled in, resulting in an increased similarity. Most examples are successful cases, in terms of similarity being increased in duplicate pairs (DPL), whilst decreasing in non-duplicate pairs (NDPL). Exceptions to this are the examples of the NORMALIZE ADDRESS and ACRONYMIZE preparators, where DPL pairs' similarities decrease and NDPL pairs' similarities increase. First, regarding the NORMALIZE ADDRESS, this effect does not necessarily lead to worse overall results. The reason for this is that we consider the preparation beneficial if it is easier to separate the pairs afterwards, even if the NDPL pairs' similarity increases. Thus, the relative similarity of all DPL and NDPL pairs is more important, such as in the case of Hotels, where the DPL pair's similarity increases more than the NDPL pair's. The importance of relative similarity is considered in our candidate selection formula in Phase 3 of Section 5. Second, the examples of ACRONYMIZE represent cases that lead to worse results, for both DPL and NDPL pairs, and for this reason are not selected from our process.

Next, Section 6.2.1 discusses the aggregated results over *preparators*, whereas Section 6.2.2 does the same from the *datasets'* viewpoint. By combining these perspectives, we can obtain a better

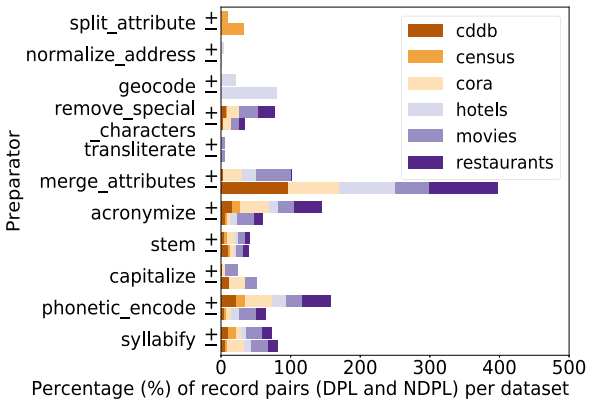

Fig. 3. Percentage (%) of record pairs (DPL and NDPL) whose similarities improved (+) or worsened (−), across *preparators* for all datasets.

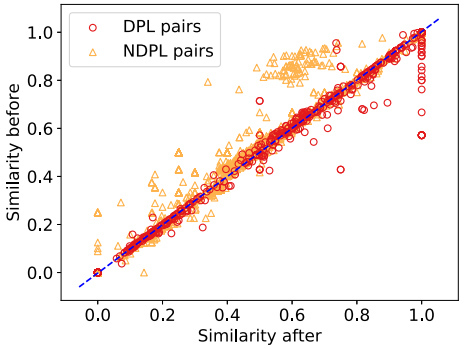

Fig. 4. Similarity difference (before and after) of preparator REMOVE SPECIAL CHARACTERS.

picture of the overall preparations' effect. We confirm their success in the final duplicate classification experiments, discussed in Section 6.3.

6.2.1 Effect of Preparators. We define the positive effect ("+") in DPL pairs to take place when they become more similar, whereas for NDPL pairs when they become less similar. Vice versa, a negative effect ("−") exists when DPL pairs become less similar and NDPL pairs more similar. Following these two definitions of positive and negative effects, in Figure 3 we show the percentage of all pairs whose similarities are affected in a positive or negative way, across datasets.

Overall, most preparators either do not improve the similarities across all datasets or they do but only marginally. More specifically, REMOVE SPECIAL CHARACTERS, ACRONYMIZE, and PHONETIC ENCODE are exceptions to the previous observation and clearly improve overall more records across datasets. The first is a general preparator, which can be applied on multiple use cases, but the last two are more tailored toward deduplication. However, there are preparators, such as MERGE ATTRIBUTES and CAPITALIZE, which in fact have worsened more pairs. However, and as mentioned, this by itself is not enough to judge whether a preparator is indeed harmful or not.

To validate the previous hypothesis and to obtain a better insight concerning the contribution of a preparator, i.e., whether it is indeed helpful or not, we show another type of visualization. In Figure 4, we consider the REMOVE SPECIAL CHARACTERS preparator, which has more of a positive effect, for visual reasons randomly downsampled to 5,000 pairs (DPL and NDPL) across all datasets and attributes. We can observe the effect on similarity, before and after preparation: pairs below the main diagonal have had their similarity increased, which we prefer for DPL pairs, and decreased above the axis, ideally mostly for NDPL pairs. In this preparator, we see that a large proportion of NDPL pairs is above the main diagonal and, similarly, a noticeable subset of DPL pairs is below the main diagonal, which both align with the positive results in Figure 3.

Examining similar figures for SYLLABIFY and CAPITALIZE (not shown), which have more negative effects in similarities, we observe that although many NDPL pairs' similarity increases slightly, the simultaneous larger increase of DPL pairs' similarity, makes it easier to separate the two classes afterwards. This effect is examined in more detail in the next section for the Cora dataset. The situation is similar for other helpful preparators, such as NORMALIZE ADDRESS, with some small negative effect, but a predominant positive effect.

In summary, examining the preparators with an overall higher positive effect reveals an easier separation of the two classes, DPL and NDPL. However, this is more of an indicator and the actual

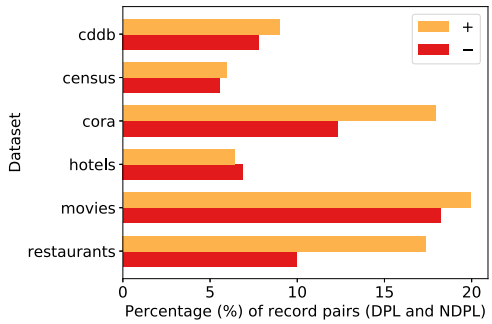

Fig. 5. Percentage (%) of record pairs (DPL and NDPL) whose similarities improved (+) or worsened (−), across *datasets*.

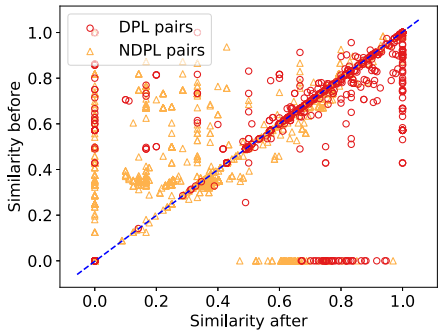

Fig. 6. Similarity difference (before and after) of dataset Cora.

positive effect of a preparator is decided in practice when it is combined with a particular dataset's attribute, as discussed in Phase 3 of Section 5.

6.2.2 Effect on Datasets. We now examine the effect that all preparations had on each dataset, w.r.t. the percentage of affected pairs. In Figure 5, we see that in fact, with the exception of the Hotels dataset, most datasets experienced an overall positive effect in similarities in the majority of the pairs.

Cora is a case with a large improvement in its pair similarities and to understand the reason behind this, we show Figure 6 to review the actual changes on the similarity of pairs. One interesting remark is that most DPL pairs have a high similarity before and after preparation, although in most cases they still managed to improve by a large degree. At the same time, there is a clear difference between most DPL and NDPL pairs, which hints good classification results. The unusual vertically aligned pairs are caused mostly by the PHONETIC ENCODE and CAPITALIZE preparators, whereas the bottom line of pairs that had 0.0 similarity before and a higher similarity after preparation are caused by preparators that populate attributes with values that were initially empty, such as MERGE ATTRIBUTES or GEOCODE and NORMALIZE ADDRESS in Hotels.

6.3 Classification Effect

Building on the previous section (Section 6.2.2), we now examine the final classification results, as discussed in Phase 5 of Section 5. In Figure 7, we see the AUC-PR scores of all datasets on the full set of DPL and NDPL pairs (gold standard). The "No preparation" and "All preparations" bars describe two baselines of (i) using the original values and (ii) applying all available valid preparations respectively. "Minimized preparations" considers the minimized set of preparations produced by our pipeline. Comparing the set of "minimized" preparations to "all" preparations serves as a good indicator to show how important is to select high quality preparations among the set of possible preparations. In practice, only a small fraction of all the available preparations were selected by our process (Table 6), using up to 6 of the 11 available preparators in total. Additionally, we present the best achievable F-measure over the precision-recall curve displayed in Figure 8. This represents the best-case scenario that a user would select to apply this process in practice, where the selected threshold of the precision-recall curve maximizes the F-measure.

Using "All preparations" does in fact improve the quality of duplicate detection in many of the datasets without the need to go through our process pipeline. An explanation for this is that a certain subset of preparators helps to improve duplicate detection in the majority of the attributes they are applied to. As an example, ACRONYMIZE and PHONETIC ENCODE are selected for many

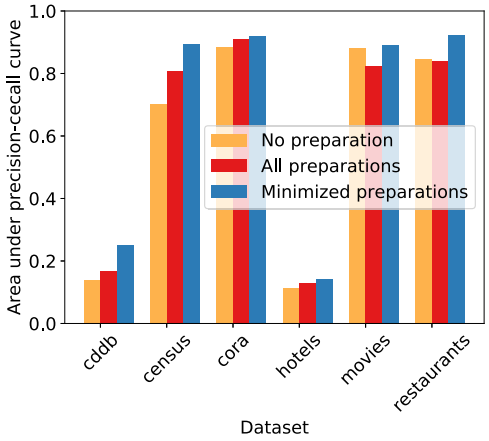

Fig. 7. Classification results according to *AUC-PR* of *original* values (left bars), prepared with *all valid preparations* (middle bars), and prepared with *minimized* set of preparations (right bars).

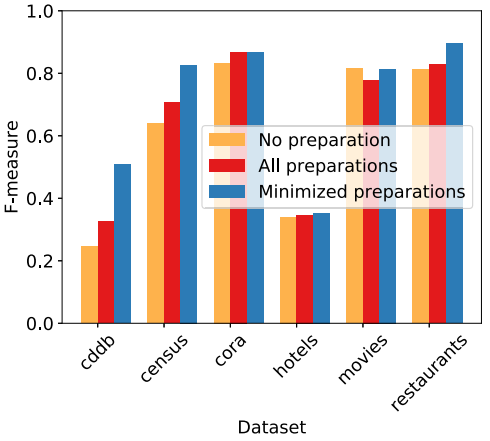

Fig. 8. Classification results according to *best achievable F-measure* of *original* values (left bars), prepared with *all valid preparations* (middle bars), and prepared with *minimized* set of preparations (right bars).

attributes by the "minimized" set of preparations, as shown in Table 6, with which we can derive their importance. However, to avoid a negative effect on the results, as is the case with AUC-PR in Movies and Restaurants, a careful selection of preparations needs to be done. The advantage of carefully selecting the preparations is verified by "Minimized preparations" bars: *All* datasets had an improvement in their classification results in AUC-PR, which is consistent with the similarity improvements (Figures 3 to 6). This is consistent as well with the leave-one-out optimization; as removing certain preparations in Phase 4 did help improve AUC-PR over the candidate preparations of Phase 3. Comparatively, with the exception of Cora in F-measure, using the minimized set of preparations achieved the best results, consistently outperforming both baselines.

In terms of improvement, CDDB, Census, and Restaurants had the largest improvements. Among them, Census achieves the largest improvement (19%), which is consistent with the positive similarity effect presented in Figure 5 in Section 6.2.2, although significantly larger. In contrast, the other three datasets did not manage to achieve such levels of improvement: value inconsistencies in Cora and Hotels are more difficult and thus not repaired by our preparations to such a degree that would make the separation of DPL and NDPL pairs easier. Movies, however, does not contain enough information in the two provided attributes that could benefit substantially from preparations. In contrast, Census consists of five different types of information (Section 4.1) that are in a more normalized format, which is more adequate to distinguish two different entities. In most of the datasets AUC-PR is increased due to improvements in precision, except for Cora that bases its better results on recall. Last, we observe that increases in AUC-PR are also reflected in the best achievable F-measures, with the exception of movies. Although there is a positive effect in both similarities and AUC-PR, this does not result in a better F-measure.

7 CONCLUSION

This article set out to analyze the impact of various data preparation activities on the success of duplicate detection. While some kind of preparation is usually vaguely reported on, it is consequential precisely which transformations were in fact executed. Our systematic analysis reveals the effects of various, selected data preparations and highlights the need for a more formal

specification of data preparation steps and the benefit of a data preparation framework, which would free data experts to repeatedly build new scripts for old or new datasets. Ideally, these preparators should be offered by a library, as those we discuss in Section 3. However, in our case the offering was not adequate, and we implemented our preparators ourselves. Overall, such a library should help developers prepare their data faster for their use cases, without the need for deeper knowledge, often provided only by domain experts.

As future work, we plan to examine the effect of a larger pool of preparators applied to more datasets. Another possible direction is to focus on different subsets of the dataset and not apply the preparator to all records but only to those that can benefit from it—conditional preparations. Finally, finding the optimal order of preparations' application that we discuss in Phase 3 of Section 5 can be difficult if time efficiency is important but may help improve the results even further.

APPENDICES

A SELECTED BLOCKING SCHEME FOR NON-DUPLICATES GENERATION

In Table 5, we present the blocking scheme used to generate the non-duplicate pairs of Section 6.1.1. Across datasets, we selected attributes that have high uniqueness, which tends to characterize better the entities. This results in smaller blocks that contain only the really similar candidates. The following strategies are used: (1) "Complete value" that uses the whole value of the record's attribute as a blocking key, (2) word n-grams that considers n consecutive words tokenized by white space of every record's attribute value, and, finally, (3) character n-grams that, similarly to word n-grams, considers n consecutive characters from every record's attribute value. In cases where the lengths of alphanumeric values are less than or equal to n the complete value is returned. Finally, the strategy and the size of n-grams (n) are selected according to the mean attribute's length, i.e., we prefer character n-grams for attributes with shorter lengths, whereas word n-grams for attributes with longer lengths.

Table 5. Blocking Scheme, Used to Generate the Non-duplicates (NDPL), as Discussed in Section 6.1.1

dataset	attribute	blocking strategy
cddb	artist	complete value
	title	word n-grams: n=6
	tracks	word n-grams: n=6
census	text	word n-grams: n=2
cora	authors	character n-grams: n=6
	title	character n-grams: n=6
hotels	hotel_name	character n-grams: n=10
	street_address1	character n-grams: n=10
movies	title	word n-grams: n=3
restaurants	name	character n-grams: n=6
	address	character n-grams: n=6
	phone	character n-grams: n=8

B PREPARATIONS PRODUCED BY EXPERIMENTS

In Table 6, we present the minimized set of preparations, as they are produced by the leave-one-out minimization of Phase 4 in Section 5. Attributes and preparators not presented in the table are not part of the minimized set of preparations. The preparators PHONETIC ENCODE and ACRONYMIZE have

been selected for many attributes. As explained in Section 5, they are both "lossy" preparators, in the sense that important information is lost. However, they do help improve duplicate detection results, which is why they are selected in many cases.

Table 6. Minimized Preparations of Experiments in Section 6

dataset	attribute	split_attribute	normalize_address	remove_special_characters	transliterate	acronymize	capitalize	phonetic_encode	syllabify	stem
cddb	artist					X	X	X		
	title					X		X		
	tracks			X		X		X	X	
census	first_name	X								
	text							X		
cora	address			X		X	X	X	X	
	authors					X		X		
	date			X		X				
	title					X		X		
hotels	city		X							
	hotel_name							X		
	street_address1		X							
movies	actors			X						
	title			X	X		X		X	
restaurants	address			X		X		X	X	
	name					X		X		X

Only preparations that are produced by our process are presented, i.e., for the attributes not presented no preparation is selected.

ACKNOWLEDGMENTS

We thank Nichole Haas and Thomas Tomosky for their valuable input on this project.

REFERENCES

[1] Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. 2016. Detecting data errors: Where are we and what needs to be done? *Proc. VLDB Endow.* 9, 12 (2016), 993–1004.

[2] Ziawasch Abedjan, John Morcos, Michael N. Gubanov, Ihab F. Ilyas, Michael Stonebraker, Paolo Papotti, and Mourad Ouzzani. 2015. DataXFormer: Leveraging the web for semantic transformations. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR'15)*.

[3] Akiko Aizawa and Keizo Oyama. 2005. A fast linkage detection scheme for multi-source information integration. In *Proceedings of the International Workshop on Challenges in Web Information Retrieval and Integration (WIRI'05)*. 30–39.

[4] Rohan Baxter, Peter Christen, and Tim Churches. 2003. A comparison of fast blocking methods for record linkage. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD'03)*, Vol. 3. 25–27.

[5] Omar Benjelloun, Hector Garcia-Molina, David Menestrina, Qi Su, Steven Euijong Whang, and Jennifer Widom. 2009. Swoosh: A generic approach to entity resolution. *VLDB J.* 18, 1 (2009), 255–276.
[6] Amit Chandel, Oktie Hassanzadeh, Nick Koudas, Mohammad Sadoghi, and Divesh Srivastava. 2007. Benchmarking declarative approximate selection predicates. In *Proceedings of the International Conference on Management of Data (SIGMOD'07)*. 353–364.
[7] Peter Christen. 2008. Automatic record linkage using seeded nearest neighbour and support vector machine classification. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD'08)*. 151–159.
[8] Peter Christen. 2008. Febrl: A freely available record linkage system with a graphical user interface. In *Proceedings of the Australasian Workshop on Health Data and Knowledge Management (HDKM'08)*. 17–25.
[9] Peter Christen. 2012. *Data Matching—Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer Data-Centric Systems and Applications.
[10] Tim Churches, Peter Christen, Kim Lim, and Justin Xi Zhu. 2002. Preparation of name and address data for record linkage using hidden Markov models. *BMC Med. Inf. Dec. Making* 2, 1 (2002), 9.
[11] Munir Cochinwala, Verghese Kurien, Gail Lalk, and Dennis Shasha. 2001. Efficient data reconciliation. *Inf. Sci.* 137, 1 (2001), 1–15.
[12] 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.
[13] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The Elements of Statistical Learning* (1st ed.). Springer-Verlag, Berlin.
[14] Sumit Gulwani. 2011. Automating string processing in spreadsheets using input-output examples. *Proceedings of the International Conference on Programming Languages (SIGPLAN)*. 317–330.
[15] Jiawei Han, Jian Pei, and Micheline Kamber. 2011. *Data Mining: Concepts and Techniques*. Elsevier.
[16] David Hand and Peter Christen. 2018. A note on using the F-measure for evaluating record linkage algorithms. *Stat. Comput.* 28, 3 (2018), 539–547.
[17] Yeye He, Xu Chu, Kris Ganjam, Yudian Zheng, Vivek Narasayya, and Surajit Chaudhuri. 2018. Transform-data-by-example (TDE): An extensible search engine for data transformations. *Proc. VLDB Endow.* 11, 10 (2018), 1165–1177.
[18] Thomas N. Herzog, Fritz J. Scheuren, and William E. Winkler. 2007. *Data Quality and Record Linkage Techniques*. Springer-Verlag, New York.
[19] James Inman. 1849. *Navigation and Nautical Astronomy, for the Use of British Seamen*. F. & J. Rivington.
[20] Zhongjun Jin, Michael R. Anderson, Michael Cafarella, and H. V. Jagadish. 2017. Foofah: Transforming data by example. In *Proceedings of the International Conference on Management of Data (SIGMOD'17)*. 683–698.
[21] Sean Kandel, Jeffrey Heer, Catherine Plaisant, Jessie Kennedy, Frank van Ham, Nathalie Henry Riche, Chris Weaver, Bongshin Lee, Dominique Brodbeck, and Paolo Buono. 2011. Research directions in data wrangling: Visualizations and transformations for usable and credible data. *Inf. Vis.* 10, 4 (2011), 271–288.
[22] Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2010. Evaluation of entity resolution approaches on real-world match problems. *Proc. VLDB Endow.* 3, 1–2 (2010), 484–493.
[23] Andreas Kunft, Asterios Katsifodimos, Sebastian Schelter, Tilmann Rabl, and Volker Markl. 2017. Blockjoin: Efficient matrix partitioning through joins. *Proc. VLDB Endow.* 10, 13 (2017), 2061–2072.
[24] Yang W. Lee, Leo L. Pipino, James D. Funk, and Richard Y. Wang. 2009. *Journey to Data Quality*. The MIT Press.
[25] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2014. *Mining of Massive Datasets*. Cambridge University Press.
[26] Pei Li, Xin Luna Dong, Andrea Maurino, and Divesh Srivastava. 2011. Linking temporal records. *Proc. VLDB Endow.* 4, 11 (2011), 956–967.
[27] Willi Mann, Nikolaus Augsten, and Panagiotis Bouros. 2016. An empirical evaluation of set similarity join techniques. *Proc. VLDB Endow.* 9, 9 (2016), 636–647.
[28] 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.
[29] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep learning for entity matching: A design space exploration. In *Proceedings of the International Conference on Management of Data (SIGMOD'18)*. 19–34.
[30] Felix Naumann and Melanie Herschel. 2010. *An Introduction to Duplicate Detection*. Morgan & Claypool Publishers.
[31] Paulo Oliveira, Fátima Rodrigues, Pedro Henriques, and Helena Galhardas. 2005. A taxonomy of data quality problems. In *Proceedings of the International Workshop on Data and Information Quality*. 219–233.
[32] Dorian Pyle. 1999. *Data Preparation for Data Mining*. Vol. 1. Morgan Kaufmann.
[33] Erhard Rahm and Hong Hai Do. 2000. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.* 23, 4 (2000), 3–13.
[34] Thomas C. Redman. 2001. *Data Quality: The Field Guide*. Digital Press.

[35] Sadhan Sood and Dmitri Loguinov. 2011. Probabilistic near-duplicate detection using simhash. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM'11)*. 1117–1126.
[36] Dinusha Vatsalan, Ziad Sehili, Peter Christen, and Erhard Rahm. 2017. Privacy-preserving record linkage for big data: Current approaches and research challenges. In *Handbook of Big Data Technologies*. Springer, 851–895.
[37] Jiannan Wang, Guoliang Li, Jeffrey Xu Yu, and Jianhua Feng. 2011. Entity matching: How similar is similar. *Proc. VLDB Endow.* 4, 10 (2011), 622–633.
[38] Y. Y. R. Wang, R. Y. Wang, M. Ziad, and Y. W. Lee. 2001. *Data Quality*. Springer US.
[39] Melanie Weis, Felix Naumann, Ulrich Jehle, Jens Lufter, and Holger Schuster. 2008. Industry-scale duplicate detection. *Proc. VLDB Endow.* 1, 2 (2008), 1253–1264.
[40] Ying Yang, Niccolo Meneghetti, Ronny Fehling, Zhen Hua Liu, and Oliver Kennedy. 2015. Lenses: An on-demand approach to ETL. *Proc. VLDB Endow.* 8, 12 (2015), 1578–1589.

Received February 2019; revised December 2019; accepted January 2020