Text similarity measures in a data deduplication pipeline for customers records

industrial experience report

Witold Andrzejewski
Poznan University of Technology
Poznan, Poland
witold.andrzejewski@cs.put.poznan.pl

Bartosz Bębel
Poznan University of Technology
Poznan, Poland
bartosz.bebel@cs.put.poznan.pl

Paweł Boiński
Poznan University of Technology
Poznan, Poland
pawel.boinski@cs.put.poznan.pl

Mariusz Sienkiewicz
Poznan University of Technology
Poznan, Poland
mariusz.sienkiewicz@doctorate.put.poznan.pl

Robert Wrembel
Poznan University of Technology
Poznan, Poland
robert.wrembel@cs.put.poznan.pl

ABSTRACT

Data stored in information systems are often erroneous. The most typical errors include: inconsistent, missing, and outdated values, typos as well as duplicates. To handle data of poor quality, data cleaning (a.k.a. curation) and deduplication (a.k.a. entity resolution) methods are used in projects realized by research and industry. Data deduplication is of particular challenge due to its computational complexity and the complexity of finding the most adequate method for comparing records and computing similarities of these records. The similarity value of two records is a compound value, whose computation is based on similarities of individual attribute values. To compute these similarities, multiple similarity measures were proposed in research literature and were implemented in various libraries (widely available in Python). For a given deduplication problem, a challenging task is to decide which similarity measures are the most adequate to given attributes being compared, since some similarity measures perform better than others for given characteristics of data being compared. In this paper, we report the experimental evaluation of 45 similarity measures for text values. The need to assess the measures came from a project conducted for a large financial institution in Poland. The measures were compared on five different real data sets, each of which had a different characteristic (e.g., text length, the number of words). The similarity measures were assessed (1) based on similarity values they produced for given values being compared and (2) based on their execution time. To the best of our knowledge, it is the first report that includes such a broad evaluation of a large selection of different similarity measures, on different real data sets.

KEYWORDS

data quality, entity resolution, data deduplication, data deduplication pipeline, customers records deduplication, text similarity measures, customer data, Python packages, experimental evaluation

1 INTRODUCTION

Institutions worldwide use data governance strategies to manage data collected by their day-to-day businesses. The strategies are supported by industry accepted guidelines (e.g., The Global Data Management Community - DAMA [11]) and by the most advanced state-of-the-art data management and data engineering solutions (e.g., database management systems, data quality assurance software). The biggest challenges in managing data in big institutions result from heterogeneous and distributed architectures used in such institutions. As a consequence, data within the same institution are represented by means of different data models and structures, moreover these data are stored in geographically distributed repositories. Accessing heterogeneous and distributed data is challenging. The most frequently addressed challenges include among others: performance of data integration processes [2, 16, 35], interoperability, standardization, comparability, and data quality [12, 17, 29]. Various data integration architectures were developed and are used as industry standards, e.g., data warehouses, data lakes, and polystores.

Despite applying data governance strategies and the aforementioned technologies, assuring high quality of data is challenging and still some of the collected data are faulty even in the same data repository. In the context of this paper *data faults* mainly concern: (1) *values of attributes* (features, fields) like: typing errors, missing, outdated, or wrong values as well as (2) the existence of multiple records in a system, which represent the same real world object, a.k.a. *duplicated data.*

Out of these, the most commonly found errors in data values include typos, inconsistent abbreviations, and different prefixes used, e.g., {'street', 'st.', 'str.'}, {'avenue', 'a.', Ave.'}. As stated in [26], a typing error is most likely to impact a third letter of a word; from 80% to 96% of typos include only one typo. In [27], the authors further analyze data errors and report that frequent typing errors include insertion, omission, transposition, substitution either of a single character or a few characters.

Faulty data also affect customer records, both individuals and institutions, since such data are typically entered manually into a system. Moreover, customer data like addresses, phone numbers, and last names change in time, i.e., get outdated. Faulty and outdated data cause a serious problem to institutions, since they deteriorate their reputation, result in economic loses, and increase customer dissatisfaction. Moreover, they distort analytical results.

A special case of faulty data are duplicated records. Such records are stored in repositories of probably the majority of companies worldwide. Duplicated data in a company are the results of (among others): (1) acquisitions of other companies, with their proper customers, (2) offered products, e.g., financial, which require for each product a separate customer instance in a system, (3) the imperfection of a software and processes used for data governance, which allow to create multiple instances of the same real customer, cf. [4], (4) user manual errors while entering data, (5) legacy systems used, which by design needed to store multiple instances of the same real customer.

To identify duplicates, the so-called data deduplication pipeline was proposed, cf. Section 2. It aims at efficiently comparing records in pairs: (1) to find highly probable records that represent the same real-world object and (2) to group these highly similar records. For each compared pair of records r_1 and r_2, similarity values of their selected attributes are computed. Based on the attribute similarities, an overall similarity between r_1 and r_2 is computed. To compute similarity values of records attributes, over 30 different similarity measures were proposed, cf. Section 3. For a given deduplication problem, a non-trivial task is to decide which similarity measures would be the most suitable for given attributes to be compared.

Paper contribution. In this paper, we present our findings on similarity measures used for computing similarities of customer records containing dirty data (notice that this paper is a substantial extension of our poster paper [5]). These findings are based on a *real R&D project* in the financial sector, which we are currently running, thus the paper is of category *industrial experience report*. This paper contributes experimental evaluation of similarity measures (cf. Section 4 and 5), suitable for our problem at hand.

Project goals. The first goal of this evaluation was to figure out which similarity measures offer in average the highest similarity value for pairs of text strings that are true positives, but include various types of errors. The second goal was to assess execution time of these measures. The experiments were run on real customers data, which included last names, institution names, and addresses. To the best of our knowledge, this paper is the first report to include such a broad evaluation of a large set of different similarity measures, run on real customers data sets having different characteristics.

Disclaimer. Since this paper presents findings from a real R&D project in the financial sector, most of the information about the project cannot be revealed. First, the detailed findings from the project are treated as the company know-how. Second, some of the test data that we used are sensitive according to the GDPR regulation. Third, the NDA agreement that we have signed prevents from publishing any data. For these reasons, the data cannot be made public. Finally, all our test data are Polish names and addresses, thus their values and errors are specific to one country only.

2 DATA DEDUPLICATION PIPELINE

A remedy for the problem of duplicated data is their deduplication, combined with prior data cleaning. A data deduplication is also known as entity resolution [23, 31]. In the research literature, a base-line data deduplication pipeline has been proposed [9, 10, 14, 18, 24, 25]. It has become a reference pipeline in multiple data deduplication projects.

The base-line deduplication pipeline assumes that data ingested by the pipeline were previously cleaned. In this context, cleaning consists among others in: homogenizing values of record attributes, correcting typing errors, replacing nulls with real true values, replacing abbreviations with full names, and homogenizing data formats.

The pipeline includes four basic tasks, namely:

- blocking (a.k.a. indexing) - it organizes records into groups, such that each group includes records that may include potential duplicates;
- block processing (a.k.a. filtering) - its goal is to eliminate records that do not have to be compared;
- entity matching (a.k.a. similarity computation) - it computes similarity values between records compared in pairs, i.e., a value of each attribute in one record is compared to a value of a corresponding attribute in the second record;
- entity clustering - it aims at creating groups of similar records, from pairs of records representing highly probable duplicates.

As mentioned before, cleaning records is needed prior to deduplicating them. Unfortunately, in practice it is impossible to perfectly clean all data delivered to the deduplication pipeline. First, because not all data can be cleaned automatically - in such cases an expert knowledge and manual cleaning are needed. Second, the amount of data that needs to be cleaned by a human may be too large to be cleaned within a finite time and at reasonable monetary costs.

On top of it, *in the financial sector data deduplication is challenging* for the following reasons. First, because an arbitrary full cleaning is not allowed by regulations in the sector, cf. [4]. The regulations prevent even known dirty customers data from being cleaned without an explicit permission of a customer (e.g., customer's first and last names with evident typos). For this reason, only simple cleaning is possible, like removing leading or trailing erroneous characters (e.g., dots, dashes, commas, white spaces) from customers addresses. Second, since customers data cannot be fully cleaned, a deduplication process has to work on only partially cleaned data. This in turn, impacts the quality of the deduplication process, since dirty records are compared and their similarities are computed based on dirty data. We reported on this and other problems related to data cleaning and deduplication in [4, 30], based on the same project as referred to in this paper.

Having said that, in commercial projects only partially cleaned data are delivered to the deduplication pipeline. Thus, it is obvious that the existing errors may substantially decrease the similarity of compared strings in the deduplication pipeline. Moreover, a similarity value between two character values varies depending on a similarity measure applied. For example, the value of a similarity measure between *'Avenida de la Reina María Cristina'* and *'ave. de la reina M.Cristina'* depends on a measure used. A few such cases are listed in Table 1, where we show the values of 6 similarity measures for these two example street names. These measures are available in different Python packages, whose names are also listed in the table. As we can see, the similarity values may vary substantially, cf. the values of measures *Overlap* and *2-gram*.

Notice that, these two example street names represent uncleaned true positives, i.e., they refer to the same real street. As such, they should be treated as the same entities, i.e., their similarity value should be as high as possible. To this end, one should use a similarity measure returning the highest value. In our example it is *Overlap* available in package *textdistance*.

For text data, multiple similarity measures have been proposed, cf. Section 3. Their inventors point out the most suitable application scenarios for the measures. For example, Hamming

Table 1: Example values of some similarity measures for two true positive street names: *'Avenida de la Reina María Cristina'* and *'ave. de la reina M.Cristina'*

similarity measure	package	value
Levenshtein	strsimpy	0.676
Jaro	jellyfish	0.740
Jaccard	textdistance	0.694
1-gram	ngram	0.694
2-gram	ngram	0.465
Overlap	textdistance	0.926

similarity measure was developed to compare equi-length strings, Overlapp was proposed to handle letter transpositions, whereas Jaro-Winkler returns higher similarities for character strings that match from the beginning. However, a real and not yet fully addressed **problem** is to figure out which of the measures return in overall a very high similarity value for a mixture of value errors (which typically exists in a given data set being deduplicated).

In an ideal scenario, one would like to use a single similarity measure in her/his entity matching task, which would return the highest similarity value for given two true positive values, even if these values were not identical, due to typing errors and different abbreviations used. To the best of our knowledge, the problem of selecting the most suitable similarity measure for a given data set to be deduplicated hasn't been addressed yet by the research community. Moreover, a comprehensive evaluation of a large number of similarity measures has not been made available either.

3 SIMILARITY MEASURES FOR TEXT DATA

The literature on data deduplication and similarity measures lists well over 30 different similarity measures for text data. One may find in the literature suggestions supported by experimental evaluations on the applicability of different measures to different text data, e.g., [1, 6, 7, 13], but they focus on small sets of measures and on small varieties of test data, as contrasted with the evaluation included in this paper.

The most frequently used similarity measures for text data are typically categorized as [8, 22]:

- based on a *phonetic encoding* of a text value, e.g., Soundex, Phonex, Phonix, NYSIIS, Metaphone;
- based on an *edit distance*, which counts the smallest number of edit operations that are required to convert string s_1 into s_2, e.g., Levenshtein, Damerau-Levenshtein, Smith-Waterman;
- based on *n-grams*, where each compared string s_1 and s_2 is split into n-character sub-strings, i.e., n-grams, and then n-grams common to both strings are counted;
- based on set similarity, e.g., Overlap, Jaccard, Sorensen-dice, where a similarity depends on the number of common elements in compared sets;
- based on either *the longest common sub-sequence* or *the longest common sub-string*;
- based on a *vector representation* in an m-dimensional space, e.g., TFIDF, Cosine;
- based on popular *compression* techniques, e.g., BZ2 (based on bzip2), LZMA (based on Lempel–Ziv–Markov chain), ZLib (based on the DEFLATE algorithm);
- *ensemble* of measures, like (1) Monge-Elkan combined with Damerau-Levenshtein and combined with n-gram or (2) Cosine combined with n-gram.

Detailed and informative descriptions of various similarity measures for text data can be found in [1, 6–8, 13, 15, 21, 22, 32].

4 EXPERIMENTAL SETUP AND PROTOCOL

In this section we describe our experimental environment, the tested similarity measures, our test data sets, and our procedure for running the evaluation.

Our procedure for assessing the similarity measures is of type unsupervised learning, as we do not build any trainable model. We run experiments using the aforementioned data set and collect similarity values returned by the measures, with a final goal to identify these measures that on average (for each data set) return the highest similarity values for compared text strings, which represent true positives. Such similarity measures are then used in one of the next steps in our deduplication pipeline for computing an overall similarity value within pairs of compared records.

An alternative approach was proposed in [19], but for the purpose of finding equivalent similarity measures, thus their ranking was important but not similarity values returned by the measures. To this end, the authors applied a supervised learning, i.e., a binary classifier. Such a classifier predicted whether two measures applied to the same character string could be treated as equivalent. The measures were applied to binary as well as numerical data. In [3], the authors also proposed a binary classifier to learn the most appropriate text similarity measures for a given deduplication problem at hand. In contrast to [19], in our application scenario, a similarity value is of primary importance as in one of the next steps in our deduplication pipeline, similarity values are used to build a rule system for deciding about similarities of whole customers records.

A supervised learning approach, similar to [19] or to [3], theoretically could work for our problem as well, provided that a sufficiently large training data set was available. In both of these approaches small training data sets were used (of not more than 1900 data items). In our case, creating a training data sets for over 11 million of customers records, where for each record over 20 attributes are compared by similarity measures, would be too time costly. For this reason, we opted for an unsupervised approach.

4.1 Choosing experimental environment

In the project, two production environments are available. The first one is a typical *data engineering environment* that includes: (1) a popular commercial RDBMS to store data and (2) SQL and an in-database procedural language to process data at the back-end. The second one is a typical *data science environment* that includes: (1) csv files to store input data, (2) Python programs to process data at the back-end, (3) spreadsheet files to store intermediate results, and (4) SQLite database to store data produced by the data deduplication pipeline.

Even though the discussion in [28] clearly shows the advantages of using the data engineering environment for data processing, as compared to the data science environment, in this project we decided to use the data science environment for the following reasons. First, the available RDBMS does not support ready to use algorithms for tasks 3 and 4 in the base-line data deduplication pipeline, i.e., few similarity measures are available out-of-the-box and no algorithms for record matching are available. Second, a pilot performance evaluation of both environments showed that the *data science environment* was faster than the *data engineering environment*. For these reasons, the *data science environment* was selected as a deployment environment. It must be stressed that the performance evaluation is valid only for the particular IT infrastructure used by the company and must not be generalized, however, it gives some valuable insights on the performance of the tested similarity measures.

Finally, all experiments described in this section were run on a workstation equipped with 16GB of RAM, Intel Core i5-8350 1.7GHz, and Samsung MZVLB256HBHQ-000L7 SSD, run under Windows10 enterprise.

4.2 Choosing data sets

The experiments were run on five different real data sets, obtained from profiling customers data. Notice that the data sets stored text values in Polish.

The data sets included:

- *customers last names* composed of *one word* - they represent short strings; this set includes 754 rows; maximum length of 1-word last names is 14 characters (on average 10.6 characters);
- *customers last names* composed of *two words* - they represent medium length strings; 419 rows; maximum length of 2-word last names is 26 characters (on average 21.2 characters);
- a *mixture* of 1-word (98%) and 2-word last names (2%), which reflects a real distribution of such last names in the customer database of the company; 782 rows; maximum length of a last name is 22 characters (on average 10.9 characters);
- *addresses* (street names) - they represent medium length strings; 1115 rows; maximum length of street names is 28 characters (on average 21.2 characters);
- *institution names* - they represent long strings; 1300 rows; maximum length of institution names is 116 characters (on average 45.4 characters).

Notice that all these test data represented true positives, but with typical real errors. Having profiled real customers data, e.g., first and last names, company names, addresses, we conclude that such data typically include: typos, missing letters, additional letters inserted, special characters inserted, transposed multiple letters in one word, multiple inconsistent abbreviations. Such errors were present in the test data sets as well.

The values in the aforementioned data sets included typical errors found by profiling customers data, i.e., letter omissions, letter transpositions, the lack of Polish diacritical signs (e.g., ą replaced by a, ę replaced by e, ł replaced by l, ń replaced by n). For street names and institution names the inconsistencies included additionally word transpositions and various abbreviations of the same word. Types of companies were written in multiple ways; some examples of naming type 'limited liability company' are shown in Table 2.

Table 4 summarizes basic data characteristics of our test data sets, like: (1) avg, min, and max number of characters, (2) avg, min, max number of words, and (3) a number of rows in a given data set.

4.3 Choosing similarity measures

The experimental evaluations and findings reported in the research literature [1, 6, 7, 13] were only a starting point for the evaluation on the applicability of similarity measures to text data,

Table 2: Examples of inconsistencies in naming 'limited liability company'; correct forms are: SP. Z O. O. or SPÓŁKA Z O. O.

incorrect forms
SPOLKA Z O O
SP. Z OGR. ODPOWIEDZIALN.
SP. Z OGRANICZONĄ ODP.
SP ZOO
SPÓŁKA Z OGR. ODP.
SPZOO
SP. z OGRANICZ. ODPOWIEDZIALNOSCIĄ

reported in this paper. Notice that, in our project features describing customers include mainly text strings with Polish letters. Therefore, similarity measures based on a phonetic encoding turned out to be inadequate in general.

We evaluated 45 measures available in the following Python packages: *distance*, *textdistance*, *strsimpy*, *jellyfish*, *nltk*, *ngram*, *difflib*, *fuzzywuzzy*, cf. Appendix. Some measures, e.g., Levenshtein, Jaro, Jaro-Winkler, Jaccard, Sorensen, are available in a few different packages, thus we evaluated these different implementations as well.

Selecting similarity measures for evaluation was based on the following criteria: (1) the popularity of the measure, based on conclusions from the following publications: [1, 6–8, 22], (2) preliminary evaluation of multiple similarity measures in a small pilot project, run prior to the main experiment. In the pilot project we evaluated over 70 different basic and ensemble measures. Based on the results, we selected measures that returned similarity values above a defined threshold. As stated earlier, in this paper we report the results obtained for 45 measures that in our opinion offered promising results.

The following **basic** similarity measures and distances[1] were tested: Levenshtein, Damerau-Levenshtein, Jaro, Jaro-Winkler, Smith-Waterman, Jaccard, Overlap, Bag, n-gram, Cosine, Sorensen, Sorensen-dice, StrCmp95, Needleman-Wunsh, Gotoh, Tversky, Longest common sub-sequence, Longest common sub-string, Ratcliff-Obershelp, Square root, BZ2, LZMA, ZLib, Sequence-Matcher, Prefix, and Editex.

We also evaluated the so-called ensemble (compound) similarity measures, which combine at least two different similarity measures. For example, in Python package *NLTK*, the Jaccard distance is available as function *jaccard_distance*. It accepts as an input parameter the value that defines an n-gram. In package *textdistance*, the Monge-Elkan similarity measure is available as function *MongeElkan*. As the first parameter it accepts the value representing n-gram and as the second parameter it accepts a proper similarity measure. The first parameter is used to build n-grams from the compared text values. Then, these n-grams are compared using a measure provided by the second parameter.

The following **ensemble** similarity measures were evaluated:

- Jaccard combined with 1-gram, 2-gram, and 3-gram,
- Cosine combined with 1-gram, 2-gram, and 3-gram,
- Monge-Elkan, combined with Damerau-Levenshtein and n-gram,

[1]Notice that there is a transformation of a distance to its corresponding similarity value [8]

- Jaro-Winkler, Sorensen-dice, StrCmp95, Overlap, Levenshtein, Damerau-Levenshtein, each of which was combined with n-gram, where n ranged from 1 to 12. n=12 applied only to long strings, i.e., institution names. We are not presenting these detailed results in the paper, but for each data set, we show in charts the highest similarity value that was returned.
  The highest value was returned by Monge-Elkan, combined with Damerau-Levenshtein and n-gram. For addresses, n-gram means 6-gram; for institutions names n-gram means 10-gram; for 1-word names n-gram means 7-gram; for 2-word names n-gram means 4-gram.

4.4 Procedure for performing experiments

Each row in every test data set, which we used, was composed of two attributes, denoted as A_1 and A_2, which stored text values. Some of them were correct, whereas others included real errors, obtained from customers profiling, cf. Section 4.2, but both values represented the same same real world object. As such, a similarity measure applied to A_1 and A_2 for a given row should have returned a high value.

The experimental protocol we used was as follows. First, for each record in a given test data set, values of A_1 and A_2 were compared by a tested similarity measure. Second, the average similarity value of the tested measure was computed for the whole data set. These steps were repeated for all 45 tested similarity measures on the same file. Their average similarities are visualized in charts reported in Section 5.

This protocol was run separately for each of the five test data sets.

5 EXPERIMENTAL RESULTS

In this section we report similarity values obtained for the aforementioned 45 similarity measures for: (1) short strings - represented by 1-word last names, (2) medium length strings - represented by 2-word last names, (3) mixed data set composed of 1-word and 2-word last names, (4) medium length strings - represented by street names, and (5) long strings - represented by institution names.

5.1 Similarity values of short strings: 1-word last names

The obtained average similarity values for 1-word last names are shown in Figure 1.

As we can observe from the chart, the same highest average similarity, which is equal to 0.928, was returned by three measures, namely:

- Jaro-Winkler from package *strsimpy* (in the chart, on the horizontal axis it is denoted as label *STRSIM_Jar_Win*),
- Overlap from package *textdistance* (denoted as *TXTDIS_Overlap*),
- StrCmp95 from package *textdistance* (denoted as *TXTDIS_StrCm95*).

The top 3 highest similarity values are also reported in summary Table 4. Notice that, the labels of all similarity functions shown in the chart are explained in Appendix.

5.2 Similarity values of medium length strings: 2-word last names

The obtained average similarity values for 2-word last names are shown in Figure 2.

The top 3 highest similarity values were returned by the following measures:

- Overlap from package *textdistance* (denoted as *TXTDIS_Overlap*): value 0.942,
- StrCmp95 from package *textdistance* (denoted as *TXTDIS_StrCm95*): value 0.938,
- Jaro-Winkler from package *textdistance* (denoted as *TXTDIS_Jar_Win*) and Jaro-Winkler from package *jellyfish* (denoted as *JELLYF_Jar_Win*): value 0.936.

5.3 Similarity values of mixed data set: 1-word and 2-word last names

We also measured similarities of customers last names in a data set composed of 1-word and 2-word last names in the following proportion: 98% of 1-word names and 2% of 2-word names. Such a proportion reflected a real distribution of last names in our customers database.

The analysis of the obtained results revealed that they were the same as reported for 1-word names, i.e., the average similarity value was equal to 0.928 and this value was returned by the same measures and from the same packages as for 1-word last names. For this reason, the experimental results on this mixed names data set are not visualized in a chart, but they are summarized in Table 4.

5.4 Similarity values of medium length strings: street names

The obtained similarity values for street names, are shown in Figure 3.

The top 3 highest similarity values were returned by the following measures:

- Overlap from package *textdistance* (denoted as *TXTDIS_Overlap*): value 0.893,
- Jaro-Winkler from package *strsimpy* (denoted as *STRSIM_Jar_Win*): value 0.863,
- StrCmp95 from package *textdistance* (denoted as *TXTDIS_StrCm95*): value 0.856.

5.5 Similarity values of long strings: institution names

The obtained similarity values for institution names are shown in Figure 4.

The top 3 highest similarity values were returned by the following measures:

- Overlap from package *textdistance* (denoted as *TXTDIS_Overlap*): value 0.921,
- Sorensen from package *textdistance* (denoted as *DISTAN_Sorens*): value 0.861,
- StrCmp95 from package *textdistance* (denoted as *TXTDIS_StrCm95*): value 0.828.

5.6 Execution times of similarity measures

In this experiment we measured execution times of the tested implementations of similarity measures on the largest data set, i.e., containing institution names. Each function implementing a

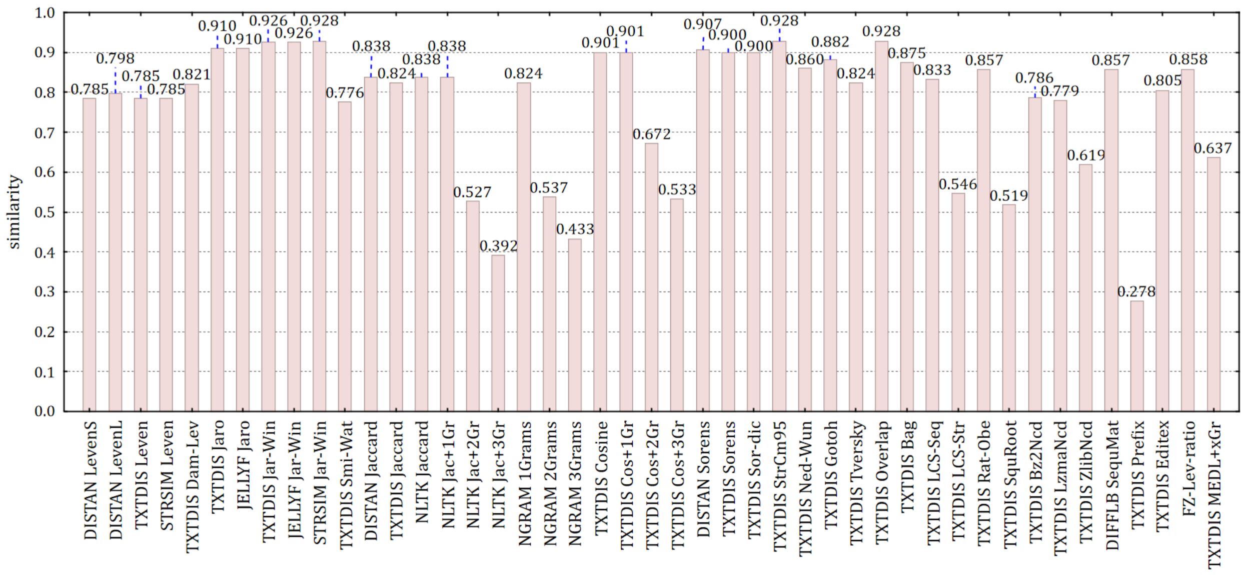

Figure 1: Average values of similarity measures for 1-word last names (short strings); max string length = 14 char, avg string length = 10.6 char, min length = 7 char

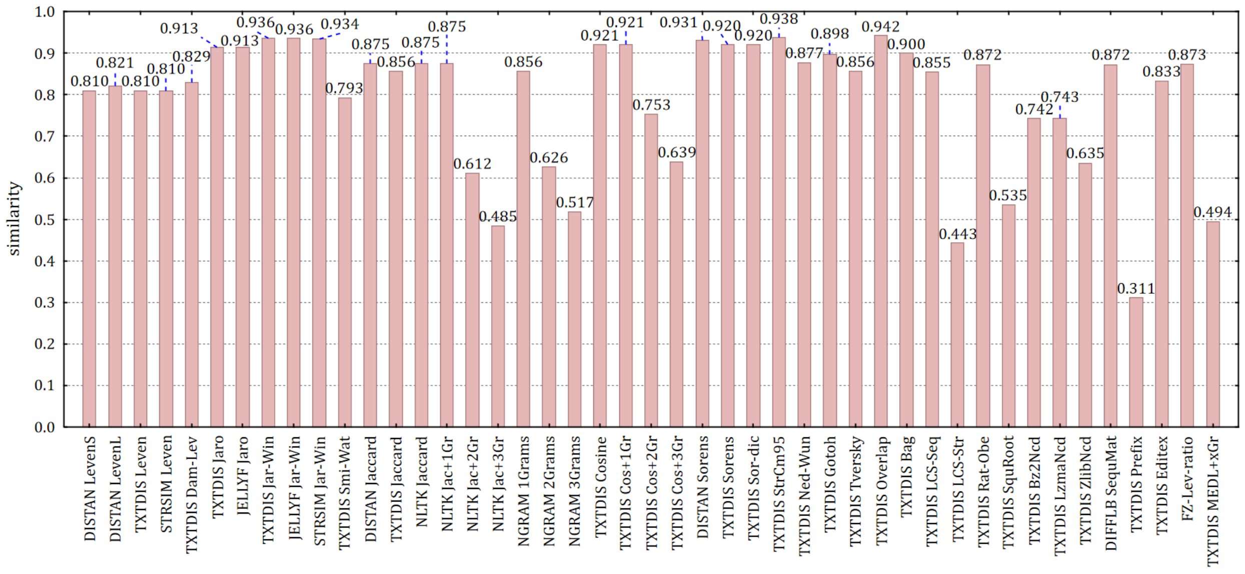

Figure 2: Average values of similarity measures for 2-word last names (medium length strings); max string length = 27 char, avg string length = 21.1 char, min length = 16 char

similarity measure was executed 12 times on the data set. The lowest and highest measurements were discarded, then the average of the remaining 10 executions was computed. Figure 5 reports the obtained average execution times. These times, measured in seconds, are shown in the Y axis on the logarithmic scale.

The chart clearly shows that some measures are terribly costly, e.g., labels *TXTDIS_LzmaNcd*, *TXTDIS_Editex*, *TXTDIS_Gotoh*. Additionally, the execution time of Monge-Elkan combined with Damerau-Levenshtein and 10-gram (labeled as *TXTDIS MEDL+xGr*) took 746 seconds. For this reason it was not included in the chart. Notice also that execution costs of some popular similarity measures, i.e., Levenshtein (labels *DISTAN_LevenS*, *DISTAN_LevenL*, *STRSIM_Leven*) can be high as compared for example to Jaro-Winkler (e.g., labels *JELLYF_Jar_Win*, *STRSIM_Jar_Win*).

Standard deviations of the measures were computed as well. In Table 3 we report their values only for the bold-blue labels on the horizontal axis in Figure 5.

Finally, it is worth to note that some attempts to improve performance of the Levenshtein, e.g., [20, 33] and Jaro-Winkler measures, e.g., [34] have been made, but their implementations are not yet available in Python packages.

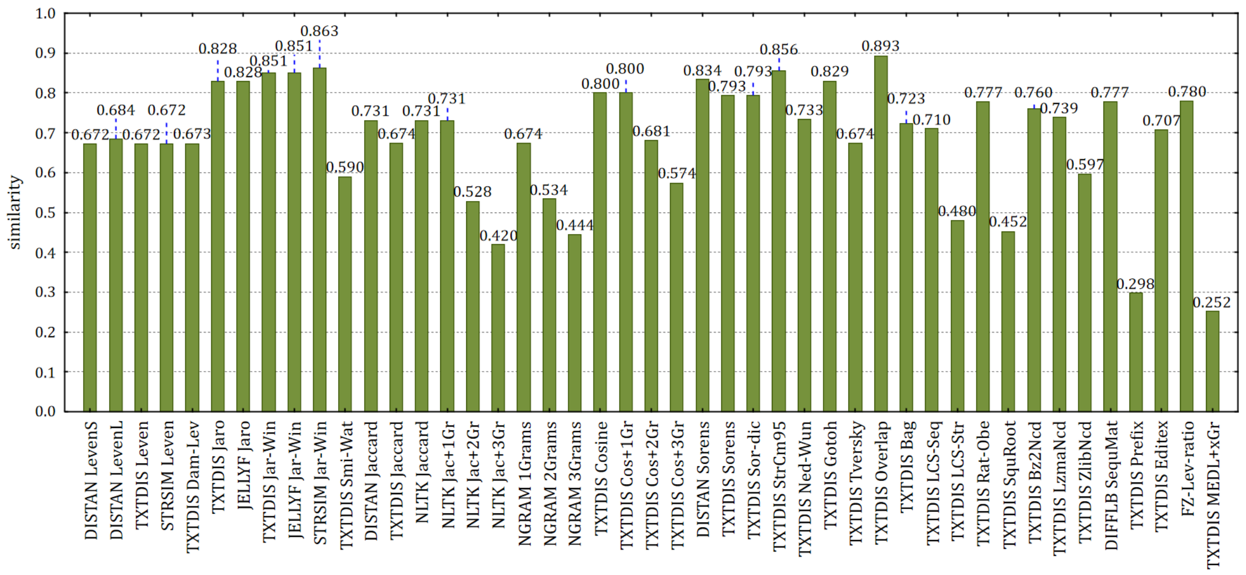

Figure 3: Average values of similarity measures for street names (medium length strings); max string length = 28 char, avg string length = 16 char, min length = 7 char

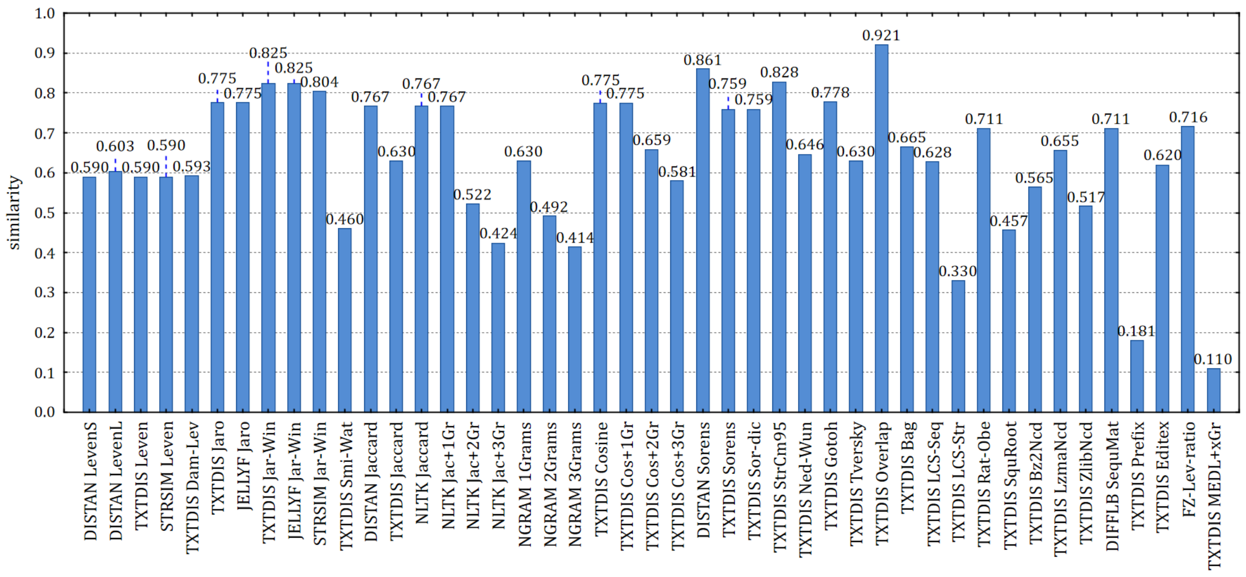

Figure 4: Average values of various similarity measures for institution names (long strings); max string length = 116 char, avg string length = 45.5 char, min length = 13 char

Table 3: Values of execution times and standard deviations of selected (the most suitable for our application scenario) similarity measures; the names of Python packages are given in parenthesis

similarity measure	exec. time [sec]	st. deviation
Jaro-Winkler (textdistance)	0.073	0.002
Jaro-Winkler (jellyfish)	0.062	0.001
Jaro-Winkler (strsimpy)	0.172	0.006
Sorensen (textdistance)	0.064	0.004
StrCmp95 (textdistance)	0.390	0.061
Overlap (textdistance)	0.108	0.004

5.7 Experiments summary

Values of execution times (labels on the horizontal axis marked bold and blue in Figure 5) indicate the similarity measures that returned the highest similarity values, when tested on long character strings. Since their execution costs are at the same time reasonably low, these measures are good candidates to be used in some deduplication projects.

The summary of the data sets used in the experiments as well as similarity measures and their values are included in Table 4. From the analysis of the charts and the content of this table we draw the following conclusions:

- for short and medium length strings, like last names and street names, composed from 7 to 28 characters, in our

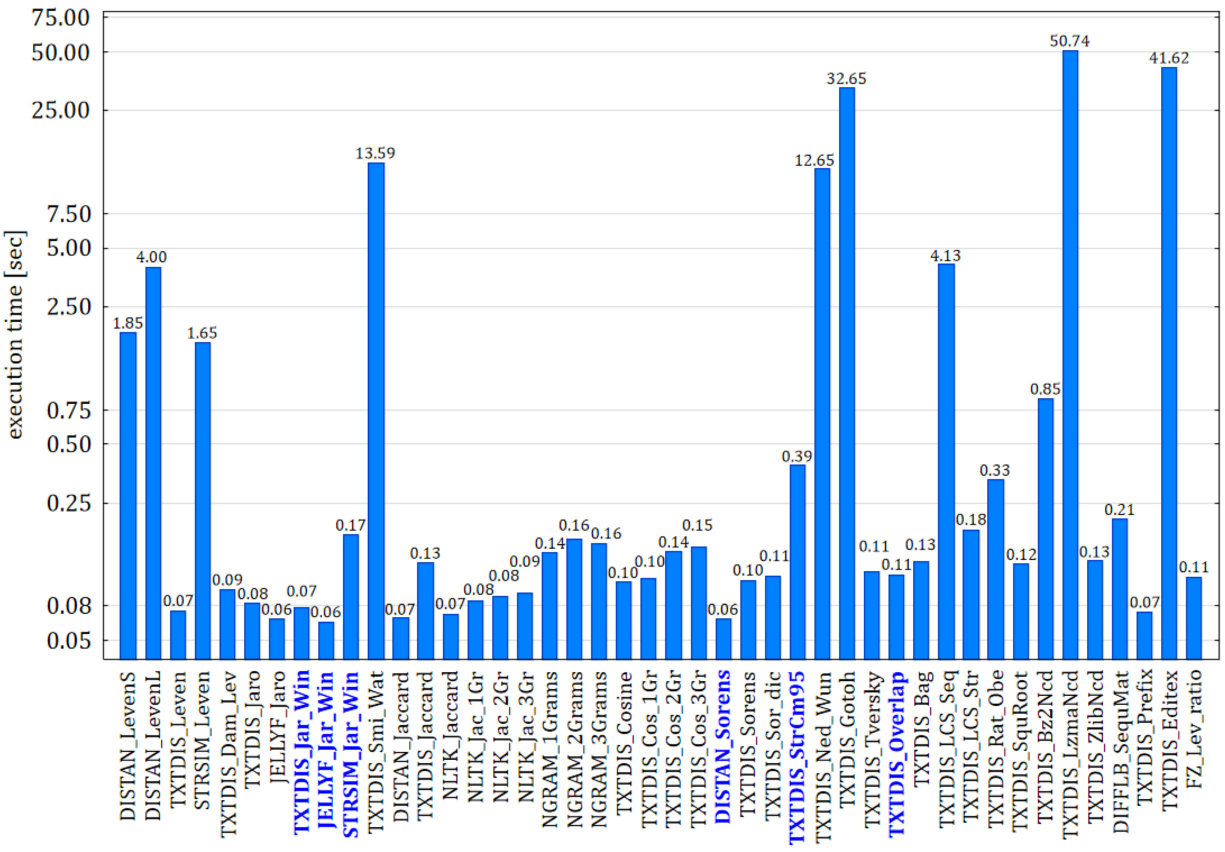

Figure 5: Execution times of similarity measures on institution names (average from 10 executions)

experiments the Overlap, Jaro-Winkler, and StrCmp95 similarity measures returned the highest similarity values, cf., column *AVG similarity* in Table 4;

- for long strings, like institution names (composed from 46 to 116 characters and up to 12 separate words) the Overlap, Sorensen, and StrCmp95 measures returned the highest similarity values.

Based on the above findings, in our project, we apply Jaro-Winkler, StrCmp95, and Overlap for variable length texts and Hamming for fixed length texts like birth dates and national ID numbers.

The above findings on the similarity measure performance should be seriously taken into consideration while deduplicating large sets of data, since measures with high execution costs will drastically deteriorate performance of the whole deduplication pipeline.

Based on these findings, it is recommended that one should apply different similarity measures to different attributes (taking into account particular data characteristics). Surprisingly, one can observe that different implementations (in different Python packages) of the same similarity measure, e.g., Levenshtein, Jaro-Winkler, or Jaccard, may return different similarity values for exactly the same pairs of character strings being compared.

6 PAPER SUMMARY

In this paper we reported experimental evaluation of 45 the most popular similarity measures for text values. To the best of our knowledge, it is the first report that includes such a broad evaluation of a large set of similarity measures on real data sets of different characteristics.

The evaluations presented in this paper were proven to be adequate to our application scenario since the findings from the experiments were incorporated in a deduplication pipeline for customers data that we implemented in the project (for details refer to [4]). The pipeline produced satisfactory results for the financial institution on a data set of over 2 million of customers. Moreover, the lessons learned from the experiments constitute a knowledge base of the project. The final goal of the described project is to apply the deduplication pipeline to a data set of customers composed of over 11 million of records.

The aforementioned project and the described experiments allowed us to identify some **future directions** towards improving the base-line data deduplication pipeline.

- First, in each of the four steps (cf. Section 2) multiple alternative algorithms may be used. For this reason, a guidance by means of an 'intelligent' software is needed in order to help a designer to choose the most adequate algorithms for a given data set at hand.
- Second, there is no method that would allow to automatically or semi-automatically choose a suitable similarity measure to a given data set (characterized by character string lengths, types of errors and their distribution).
- Third, there are no tools that would assist a designer in choosing and testing similarity measures and selecting the most adequate one to a given application scenario.

Table 4: Values of top 3 similarity measures for different characteristics of text data (short, medium, and long character strings); AVG(len), MIN(len), and MAX(len) are measured in characters; the names of Python packages are given in parenthesis

Data	Characteristics	Sim measure (package)	AVG(sim)
Last names (1-word)	AVG(len)=10.6 MIN(len)=7 MAX(len)=14 754 rows	Jaro-Winkler (strsimpy) Overlap (textdistance) StrCmp95 (textdistance)	0.928 0.928 0.928
Last names (2-words)	AVG(len)=21.2 MIN(len)=16 MAX(len)=27 419 rows	Overlap (textdistance) StrCmp95 (textdistance) Jaro-Winkler (textdistance) Jaro-Winkler (jellyfish)	0.942 0.938 0.936 0.936
Last names (mixed)	AVG(len)=10.9 MIN(len)=7 MAX(len)=22 98% of 1-word 2% of 2-word 782 rows	Jaro-Winkler (strsimpy) Overlap (textdistance) StrCmp95 (textdistance)	0.928 0.928 0.928
Street names	AVG(len)=16.0 MIN(len)=7 MAX(len)=28 AVG #words=2.4 MIN #words=1 MAX #words=4 1115 rows	Overlap (textdistance) Jaro-Winkler (strsimpy) StrCmp95 (textdistance)	0.893 0.863 0.856
Institution names	AVG(len)=45.5 MIN(len)=13 MAX(len)=116 AVG #words=6.3 MIN #words=1 MAX #words=12 1300 rows	Overlap (textdistance) Sorensen (textdistance) StrCmp95 (textdistance)	0.921 0.861 0.828

- Finally, our unsupervised approach to finding the most appropriate similarity measures for a given data set could be contrasted with supervised methods, similar to those used in [3, 19].

Acknowledgements. The project is supported by the grant from the National Center for Research and Development no. POIR.01.01.01-00-0287/19. The work of Mariusz Sienkiewicz is additionally supported by the Applied Doctorate grant no. DWD/4/24/2020 from the Polish Ministry of Education and Science.

REFERENCES

[1] Madhavi Alamuri, Bapi Raju Surampudi, and Atul Negi. 2014. A survey of distance/similarity measures for categorical data. In *Int. Joint Conf. on Neural Networks (IJCNN)*. IEEE, 1907–1914.

[2] Syed Muhammad Fawad Ali and Robert Wrembel. 2017. From conceptual design to performance optimization of ETL workflows: current state of research and open problems. *VLDB Journal* 26, 6 (2017), 777–801.

[3] Mikhail Bilenko and Raymond J. Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*. ACM, 39–48.

[4] Paweł Boiński, Mariusz Sienkiewicz, Bartosz Bębel, Robert Wrembel, Dariusz Gałęzowski, and Waldemar Graniszewski. 2022. On Customer Data Deduplication: Lessons Learned from a R&D Project in the Financial Sector. In *Workshops of the EDBT/ICDT 2022 Joint Conference (CEUR Workshop Proceedings)*, Vol. 3135. CEUR-WS.org.

[5] Paweł Boiński, Mariusz Sienkiewicz, Robert Wrembel, Bartosz Bębel, and Witold Andrzejewski. 2023. On evaluating text similarity measures for customer data deduplication. In *The ACM Symposium on Applied Computing (SAC) (accepted for publication)*. ACM.

[6] Shyam Boriah, Varun Chandola, and Vipin Kumar. 2008. Similarity Measures for Categorical Data: A Comparative Evaluation. In *SIAM Int. Conf. on Data Mining (SDM)*. SIAM, 243–254.

[7] Peter Christen. 2006. A Comparison of Personal Name Matching: Techniques and Practical Issues. In *Int. Conf. on Data Mining (ICDM)*. IEEE Computer Society, 290–294.

[8] Peter Christen. 2012. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer.

[9] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. 2021. An Overview of End-to-End Entity Resolution for Big Data. *Comput. Surveys* 53, 6 (2021), 127:1–127:42.

[10] Adrian Colyer. 2020. The morning paper on An overview of end-to-end entity resolution for big data. https://blog.acolyer.org/2020/12/14/entity-resolution/.

[11] Dama International. 2017. *DAMA-DMBOK: Data Management Body of Knowledge (2nd edition)*. Technics Publications.

[12] Cinzia Daraio and Wolfgang Glänzel. 2016. Grand challenges in data integration - state of the art and future perspectives: an introduction. *Scientometrics* 108, 1 (2016), 391–400.

[13] María del Pilar Angeles and Adrian Espino-Gamez. 2015. Comparison of Methods Hamming Distance, Jaro, and Monge-Elkan. In *Int. Conf. on Advances in Databases, Knowledge, and Data Applications (DBKDA)*. 63–69.

[14] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. 2007. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 19, 1 (2007), 1–16.

[15] Sergio Jiménez, Claudia Jeanneth Becerra, Alexander F. Gelbukh, and Fabio A. González. 2009. Generalized Mongue-Elkan Method for Approximate Text String Comparison. In *Int. Conf. on Computational Linguistics and Intelligent Text Processing (CICLing) (LNCS)*, Alexander F. Gelbukh (Ed.), Vol. 5449. Springer, 559–570.

[16] Anastasios Karagiannis, Panos Vassiliadis, and Alkis Simitsis. 2013. Scheduling strategies for efficient ETL execution. *Inf. Systems* 38, 6 (2013), 927–945.

[17] Sona Karkosková. 2023. Data Governance Model To Enhance Data Quality In Financial Institutions. *Inf. Syst. Manag.* 40, 1 (2023), 90–110.

[18] Hanna Köpcke and Erhard Rahm. 2010. Frameworks for entity matching: A comparison. *Data & Knowledge Engineering* 69, 2 (2010), 197–210.

[19] Marie-Jeanne Lesot and Maria Rifqi. 2010. Order-Based Equivalence Degrees for Similarity and Distance Measures. In *Int. Conf. Computational Intelligence for Knowledge-Based Systems Design (Lecture Notes in Computer Science)*, Vol. 6178. Springer, 19–28.

[20] Robert Logan, Zoe Fleischmann, Sofia Annis, Amy Wangsness Wehe, Jonathan L. Tilly, Dori C. Woods, and Konstantin Khrapko. 2022. 3GOLD: optimized Levenshtein distance for clustering third-generation sequencing data. *BMC Bioinform.* 23, 1 (2022), 95.

[21] Alvaro E. Monge and Charles Elkan. 1997. An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records. In *Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD)*.

[22] Felix Naumann. 2013. Similarity measures. Hasso Plattner Institut.

[23] George Papadakis, Ekaterini Ioannou, Emanouil Thanos, and Themis Palpanas. 2021. *The Four Generations of Entity Resolution.* Morgan & Claypool Publishers.

[24] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. 2020. Blocking and Filtering Techniques for Entity Resolution: A Survey. *Comput. Surveys* 53, 2 (2020), 31:1–31:42.

[25] George Papadakis, Leonidas Tsekouras, Emmanouil Thanos, George Giannakopoulos, Themis Palpanas, and Manolis Koubarakis. 2019. Domain- and Structure-Agnostic End-to-End Entity Resolution with JedAI. *SIGMOD Record* 48, 4 (2019), 30–36.

[26] Joseph J. Pollock and Antonio Zamora. 1983. Collection and characterization of spelling errors in scientific and scholarly text. *Journal of the American Society for Information Science* 34, 1 (1983), 51–58.

[27] Joseph J. Pollock and Antonio Zamora. 1984. Automatic Spelling Correction in Scientific and Scholarly Text. *Commun. ACM* 27, 4 (1984), 358–368.

[28] Oscar Romero and Robert Wrembel. 2020. Data Engineering for Data Science: Two Sides of the Same Coin. In *Int. Conf. on Big Data Analytics and Knowledge Discovery (DaWaK) (LNCS)*, Vol. 12393. Springer, 157–166.

[29] Shazia Wasim Sadiq, Tamraparni Dasu, Xin Luna Dong, Juliana Freire, Ihab F. Ilyas, Sebastian Link, Renée J. Miller, Felix Naumann, Xiaofang Zhou, and Divesh Srivastava. 2017. Data Quality: The Role of Empiricism. *SIGMOD Record* 46, 4 (2017), 35–43.

[30] Mariusz Sienkiewicz and Robert Wrembel. 2021. Managing Data in a Big Financial Institution: Conclusions from a R&D Project. In *Workshops of the EDBT/ICDT 2021 Joint Conference (CEUR Workshop Proceedings)*, Vol. 2841. CEUR-WS.org.

[31] Giovanni Simonini, Luca Zecchini, Sonia Bergamaschi, and Felix Naumann. 2022. Entity Resolution On-Demand. *Proc. VLDB Endowment* 15, 7 (2022), 1506–1518.

[32] Textdistance. [n.d.]. Python package: textdistance. https://pypi.org/project/textdistance/.

[33] Andrew Todd, Marziyeh Nourian, and Michela Becchi. 2017. A Memory-Efficient GPU Method for Hamming and Levenshtein Distance Similarity. In *Int. Conf. on High Performance Computing (HiPC)*. IEEE Computer Society, 408–418.

[34] Yaoshu Wang, Jianbin Qin, and Wei Wang. 2017. Efficient Approximate Entity Matching Using Jaro-Winkler Distance. In *Int. Conf. Web Information Systems Engineering (WISE) (LNCS)*, Vol. 10569. Springer, 231–239.

[35] Robert Wrembel. 2022. Data Integration, Cleaning, and Deduplication: Research Versus Industrial Projects. In *Int. Conf. Information Integration and Web Intelligence (iiWAS) (Lecture Notes in Computer Science)*, Vol. 13635. Springer, 3–17.

APPENDIX. SIMILARITY MEASURES TESTED

The following similarity or distance measures were tested in the experiment. The below list includes: a *label* used in charts to denote the measure, the name of a *measure*, and the name of a Python *package* with its implementation.

(1) label: *DISTAN_LevenS*; measure: **Levenshtein** with shortest alignment; package: **distance**
(2) label: *DISTAN_LevenL*; measure: **Levenshtein** with longest alignment; package: **distance**
(3) label: *TXTDIS_Leven*; measure: **Levenshtein**; package: **textdistance**
(4) label: *STRSIM_Leven*; measure: **Levenshtein**; package: **strsimpy**
(5) label: *TXTDIS_Dam_Lev*; measure: **Damerau-Levenshtein**; package: **textdistance**
(6) label: *TXTDIS_Jaro*; measure: **Jaro**; package: **textdistance**
(7) label: *JELLYF_Jaro*; measure: **Jaro**; package: **jellyfish**
(8) label: *TXTDIS_Jar_Win*; measure: **Jaro-Winkler**; package: **textdistance**
(9) label: *JELLYF_Jar_Win*; measure: **Jaro-Winkler**; package: **jellyfish**
(10) label: *STRSIM_Jar_Win*; measure: **Jaro-Winkler**; package: **strsimpy**
(11) label: *TXTDIS_Smi_Wat*; measure: **Smith-Waterman**; package: **textdistance**
(12) label: *DISTAN_Jaccard*; measure: **Jaccard**; package: **distance**
(13) label: *TXTDIS_Jaccard*; measure: **Jaccard**; package: **textdistance**
(14) label: *NLTK_Jaccard*; measure: **Jaccard**; package: **nltk**
(15) label: *NLTK_Jac_1Gr*; measure: **Jaccard** + **1-gram**; package: **nltk**
(16) label: *NLTK_Jac_2Gr*; measure: **Jaccard** + **2-gram**; package: **nltk**
(17) label: *NLTK_Jac_3Gr*; measure: **Jaccard** + **3-gram**; package: **nltk**
(18) label: *NGRAM_1Grams*; measure: **1-gram**; package: **ngram**
(19) label: *TXTDIS_Bz2Ncd*; measure: **BZ2**; package: **textdistance**
(20) label: *NGRAM_2Grams*; measure: **2-gram**; package: **ngram**
(21) label: *NGRAM_3Grams*; measure: **3-gram**; package: **ngram**
(22) label: *TXTDIS_Cosine*; measure: **Cosine**; package: **textdistance**
(23) label: *TXTDIS_Cos_1Gr*; measure: **Cosine** + **1-gram**; package: **textdistance**
(24) label: *TXTDIS_Cos_2Gr*; measure: **Cosine** + **2-gram**; package: **textdistance**
(25) label: *TXTDIS_Cos_3Gr*; measure: **Cosine** + **3-gram**; package: **textdistance**
(26) label: *DISTAN_Sorens*; measure: **Sorensen**; package: **distance**
(27) label: *TXTDIS_Sorens*; measure: **Sorensen**; package: **textdistance**
(28) label: *TXTDIS_Sor_dic*; measure: **Sorensen-dice**; package: **textdistance**
(29) label: *TXTDIS_StrCm95*; measure: **Strcmp95**; package: **textdistance**
(30) label: *TXTDIS_Ned_Wun*; measure: **Needleman-Wunsch**; package: **textdistance**
(31) label: *TXTDIS_Gotoh*; measure: Gotoh; package: **textdistance**
(32) label: *TXTDIS_Tversky*; measure: Tversky; package: **textdistance**
(33) label: *TXTDIS_Overlap*; measure: Overlap; package: **textdistance**
(34) label: *TXTDIS_Bag*; measure: Bag; package: **textdistance**
(35) label: *TXTDIS_LCS_Seq*; measure: **Longest Common Subsequence**; package: **textdistance**
(36) label: *TXTDIS_LCS_Str*; measure: **Longest Common Sub-string**; package: **textdistance**
(37) label: *TXTDIS_Rat_Obe*; measure: **Ratcliff-Obershelp**; package: **textdistance**
(38) label: *TXTDIS_SquRoot*; measure: **Square root**; package: **textdistance**
(39) label: *TXTDIS_LzmaNcd*; measure: **LZMA**; package: **textdistance**
(40) label: *TXTDIS_ZlibNcd*; measure: **ZLib**; package: **textdistance**
(41) label: *DIFFLB_SequMat*; measure: **SequenceMatcher**; package: **difflib**
(42) label: *TXTDIS_Prefix*; measure: **Prefix**; package: **textdistance**
(43) label: *TXTDIS_Editex*; measure: **Editex**; package: **textdistance**
(44) label: *FZ_Lev_ratio*; measure: **Levenshtein**; package: **fuzzywuzzy**
(45) label: *TXTDIS MEDL+xGr*; measure: **Monge-Elkan** + **Damerau-Levenshtein** + **n-gram**; package: **textdistance**; the following n-gram values gave the highest similarity values: 6-gram for addresses, 10-gram for institutions names, 7-gram for 1-word names, 4-gram for 2-word names