

BeRTo: An Efficient Spark-Based Tool for Linking Business Registries in Big Data Environments

Andrea Colombo^a and Francesco Invernici^b

Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Via G. Ponzio 34, Milan, Italy
andrea1.colombo@polimi.it, francesco.invernici@polimi.it

Keywords: Record Linkage, Entity Resolution, Apache Spark, Hadoop, Big Data Integration.


Abstract: Linking entities from different datasets is a crucial task for the success of modern businesses. However, aligning entities becomes challenging as common identifiers might be missing. Therefore, the process should rely on string-based attributes, such as names or addresses, thus harming precision in the matching. At the same time, powerful general-purpose record linkage tools require users to clean and pre-process the initial data, introducing a bottleneck in the success of the data integration activity and a burden on actual users. Furthermore, scalability has become a relevant issue in modern big data environments, where a lot of data flows daily from external sources. This work presents a novel record linkage tool, BeRTo, that addresses the problem of linking a specific type of data source, i.e., business registries, containing information about companies and corporations. While being domain-specific harms its usability in other contexts, it manages to reach a new frontier in terms of precision but also scalability, as it has been built on Spark. Integrating the pre-processing and cleaning steps in the same tool creates a user-friendly end-to-end pipeline that requires users only to input the raw data and set their preferred configuration, allowing to focus on recall or precision.


1 INTRODUCTION

In an evolving data analytics landscape, connecting diverse and heterogeneous data sources is paramount for enhancing the data-driven decision-making of any organization. This task is usually referred to as record linkage or entity matching, whose common obstacle is the integration of datasets when no shared identifier exists (Herzog et al., 2007; Getoor and Machanavajjhala, 2012). In such cases, the issue is reconciling and linking disparate data records based on attributes that are not primary keys, with the aim of identifying the same real-world entity, such as customers or products. Linking those entities becomes critical to fully unlock the potential of data analytics activities (Dong and Srivastava, 2013; Christophides, 2020; Chen et al., 2018). Furthermore, the advent of big data has added a new layer of complexity to data integration activities due to the volume and the variety of data sources involved. Traditional tools have in fact become inefficient since such exercises, following the flow of data from repository sources, are performed on a daily basis (Yan et al., 2020), due to

the constant changes in the received data and the need of keep-to-date internal business processes.

Various tools have been developed over time to assist in record linkage activities, mostly general-purpose ones, which serve and assist users over different record linkage scenarios. However, such tools are, by definition, incapable of natively capturing domain-specific rules or patterns that might impact the overall success, i.e., recall and precision, of matched entities, without undergoing specific and time-consuming training, mostly based on machine learning techniques. This is also the case of integrating *business registries*, a particularly relevant problem for any statistical agency and companies, especially those operating worldwide. Business registries contain information about business entities, their legal structures, ownership details, and financial information. Such data play a central role in producing national and international statistics (Eurostat, 2024) and are critical for authorities to enable more specific analysis, such as malicious market behaviors (Ryan et al., 2020), for instance, by allowing the connection of ownership data with financial information. Linking business registries usually relies on registration numbers or other national and cross-national identifiers. However, it is not rare that different data repositories are

^a  <https://orcid.org/0000-0002-7596-8863>

^b  <https://orcid.org/0009-0002-5423-6978>

based on distinct identifiers, often internal IDs, that are not worldwide and internationally adopted (Guralnick, 2015). Thus, they do not allow a full integration to users who might want to combine such information with internal datasets or factual knowledge. In such cases, utilizing non-standard and string-based attributes for entity matching can help reconcile and create a bridge between the business registries. However, accounting for attribute-specific pre-processing is essential for improving matching performance. For instance, business and organizational names can adopt conventions, such as abbreviations, which might heavily harm and limit the precision and recall of entity matching.

In this paper, we contribute to a novel open-source Spark-based entity matching tool, BeRTo, implemented in Python, that tackles the challenge of integrating business registries by exploiting legal entities' name and address information. It combines widely used techniques for record linkage, such as *fuzzy matching*, with standard string processing and string similarity techniques. Its implementation is based on the Apache Spark paradigm (Salloum et al., 2016), thus leveraging the most recent big data processing solutions, allowing smooth use in fast-paced and big-data contexts (Shaikh et al., 2019). This tool is a domain-specific solution that focuses on a specific record linkage problem related to companies. Such nature, while limiting its usage for general-purpose tasks, allows it to reach a new frontier in the recall-precision trade-off, as it can be precise and successful, due to its domain-aware features, and scalable, thanks to its Spark implementation. Additionally, BeRTo allows users to set their preferred configuration by setting its parameters, to obtain the desired level of precision and recall without the need to perform any kind of pre-processing or cleaning of original data.

Overview. In Section 2 we discuss related works, Section 3 presents the approach and the system while in Section 4 we conduct the experiments testing our system. Section 5 concludes the paper.

2 RELATED WORK

The problem of record linkage has received much attention in the past few decades. It deals with finding the tuple pairs (a, b) that refer to the same entity, be it a person, product, or institution, between two tables or, more in general, two databases (Christen, 2012). Traditional record linkage techniques focus on connecting sets of records sharing the same schema, with numerous entity-matching algorithms being proposed (Koudas et al., 2006; Elmagarmid et al., 2007).

To address the big data context, new algorithms have been proposed based on techniques such as adaptive blocking (Bilenko et al., 2006), incremental clustering techniques (Nentwig and Rahm, 2018) or by utilizing new big data processing paradigms, which balance load among different nodes (Kolb et al., 2012b; Kolb et al., 2012a). Although such techniques are popular, little or no guidance is typically provided on selecting appropriate blockers or settings for such algorithms. Furthermore, few systems also integrate a data-cleaning workflow, with most ones leaving such a pre-processing burden on users. The data cleaning part is a critical component for the success of any of these algorithms and is one of the main bottlenecks in data integration workflows (Krishnan et al., 2016). Some systems based on machine learning techniques have been proposed to link entities, which mitigate the problem of pre-processing and data cleaning (Wang et al., 2021). Such solutions achieve great results in terms of accuracy but become easily impracticable when moving to large volumes of data (Ebraheem et al., 2018). More recent works design so-called *meta-blocking* technique, such as SparkER (Gagliardelli et al., 2019) that allow a more sophisticated LLM-based solution as Ditto (Li et al., 2020) to scale nicely. However, such a combination is hard to set and does not account for specific cleaning over raw data, which is still left to users.

Magellan (Konda et al., 2016) is one of the first ecosystems that proposes a general-purpose tool to tackle the data cleaning issue. It offers an Entity Matching (EM) system that is novel in providing how-to guides and support for the entire EM pipeline, combined with tight integration with the Python data ecosystem. With Magellan, users have been able to achieve high matching accuracy on several multiple datasets. However, its general-purpose nature still leaves a high portion of the burden on users, who need both dataset knowledge and at least data integration skills to follow the guides. Therefore, in a fast-paced, big data and problem-specific environment, e.g., connecting two data sources that are regularly updated, a domain-aware tool might be preferred over a general-purpose one. Other systems, such as *LinkageWiz*¹ and *Dedupe.io* (Forest and Eder, 2015), devote more attention to the data cleaning phase and have been designed to tackle entity matching issues; however, they suffer when moving to the big data context. For instance, *LinkageWiz* states on their website that it can process files containing up to 4-5 million records and *Dedupe.io* considers databases up to 700k rows, far from the big data paradigm that is required nowadays. More recently, *Splink* (Linacre et al., 2022) has

¹<https://www.linkagewiz.net/ListManagement.htm>

been developed for scalable deduplication and record linkage. However, its probabilistic-based approach still requires a heavy burden on users to design the most suitable combination of blocking rules and system configuration to optimize it, both in terms of execution time and final precision, with some data preprocessing also suggested.

3 METHODOLOGY AND TOOL IMPLEMENTATION

In this section, we present BeRTo², our entity-matching tool for integrating large business registries in an effective and scalable way. First, we discuss the nature of business registries and the algorithms we designed and integrated into the tool for performing data cleaning and preparing the information available in a business registry for record linkage. Then, we delve into the actual matching records strategy that has been developed and we present the precision-recall trade-off that arises from utilizing BeRTo. In Figure 1, we depict an overview of the architecture of our tool.

3.1 Business Registries

Business registries, often called also company or entity databases, are repositories that systematically record and store information about businesses. Traditionally, such repositories were maintained by authorities for their jurisdiction. However, with the rise of commercial data providers, usually on a global scale, we have experienced a rise in this kind of database, reaching also large volumes. For instance, Moody’s has its own company database, Orbis³, which is the resource for entity data and provides information on close to 462 million companies and entities across the globe, with many having detailed financial information, which is of interest to many investors and customers, be they individuals or authorities (Bajgar et al., 2020). Another example is OpenCorporates dataset⁴, which allows one to search for over 220 million companies to understand better who is behind companies, which is of interest to businesses, governments, journalists, and even researchers. Although such commercial databases provide some national and international identifiers, they are not used as primary keys, and there is no guarantee that they are not null values. For instance, the International Securities Identification Number (ISIN) code is a glob-

²The tool is available at: <https://bitly.ws/3d3aJ>

³<https://bvdinfo.com/R0/Orbis>

⁴Website: <https://opencorporates.com/>

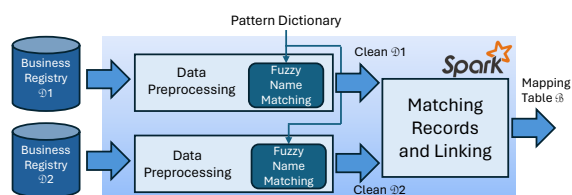


Figure 1: BeRTo’s architecture.

ally recognized identifier and can be used to identify a company through its stocks, but it is assigned only to listed companies. Therefore, many unlisted companies might not have such an identifier. In addition, commercial providers often use OCR pipelines for extracting such identifiers (Arief et al., 2018), which do not provide the same level of trustworthiness as a legally binding business registry. However, linking to such an enormous database is essential to leverage the fine-grained worldwide data commercial providers collect. A business registry usually contains data about:

- Name of the entity, i.e., the company name. It is a string attribute with multiple possible variants of the same name, i.e., abbreviations.
- Country of residence of the entity, encoded in actual names or in standard defined codes, such as ISO 3166 (alpha-2 or alpha-3 digits).
- Street address, string denoting the union of an actual street name, postal code, and city of residence. Such information might be parsed into distinct attributes.

We consider these attributes the minimum requirement that enables accurate matching across business registries. In technical terms, we consider them as our main *primary key*. While other company-specific information might be available, such as the telephone number, we consider the above as the most stable over time and spread across business registries. As we shall see, our tool also allows the use of custom attributes that might be available in the target registries. However, for such attributes, no specific preprocessing technique is foreseen.

3.2 Data Preprocessing

We developed a specific string pre-processing pipeline for each attribute we considered in the previous section, including some usual and common preliminary steps. These include accent conversion, multiple whitespaces, punctuation removal, and case-sensitive transformation (from lower to capital). For instance, an entity whose original name is *Alpha Co. Ltd* becomes *ALPHA COMPANY LTD*. Then, for each

attribute, specific preprocessing and cleaning methods have been implemented.

Name-Specific Processing. Company names are characterized by the presence of an entity-specific string followed by the corresponding business entity string, which denotes the business type. For instance, the LTD in *ALPHA COMPANY LTD* denotes that *ALPHA COMPANY* is a Private Limited Company. In many cases and jurisdictions, for such specification, if present, many variations are allowed, most importantly *abbreviations*. Therefore, to increase the matching chances between two business registries, entity names must be pre-processed to align such cases. To achieve that, our tool implements an *empirical fuzzy name matching* system, which can capture a set of the most relevant and commonly used abbreviations and variations that affect entity names. The system is described by Algorithm 1.

Algorithm 1: Empirical Fuzzy Name Matching.

Data: $\mathcal{D}_1, \mathcal{D}_2$ business registries
Input: y_1 and y_2 common ID, T threshold
Output: Fuzzy Pattern Rules

- 1 **Function** FuzzyPattern(\mathcal{Z}):
- 2 $\mathcal{Z} = \mathcal{Z} \cup (\rho(Z_{name_2}, Z_{name_1} \leftarrow Z_{name_1}, Z_{name_2}) \mathcal{Z})$
- 3 $Z_{patterns} = ndiff(Z_{name_1}, Z_{name_2})$
- 4 $P = \mathcal{Z}.GroupBy(name_1, name_2, patterns)$
- 5 $P_f = \frac{P.count()}{length(\mathcal{Z})}$
- 6 **return** P
- 7 **Function** Main:
- 8 $\mathcal{Z} \leftarrow \mathcal{D}_1 \bowtie_{y_1=y_2} \mathcal{D}_2$
- 9 $K = FuzzyPattern(\mathcal{Z}).OrderBy(P_f)$
- 10 **if** HumanEvaluation = False **then**
- 11 | $K = \sigma_{f>T} K$
- 12 **return** K

Essentially, our tool contains a sub-module run over a set of *safe matches*, obtained by linking business registries whose foreign key was available (y_1 and y_2). We integrated an *ndiff* function on such linked entities, which returns the string difference between two strings. The returned differences $Z_{patterns}$ define *string patterns*, i.e., sub-strings that do not allow to join entities directly using the name. For instance, if $Z_{name_1} = corp$ and $Z_{name_2} = corporation$ then $Z_{patterns} = oration$. Then the frequency of the patterns is computed and, whenever the relative frequency is high, the pattern can be added to a *pattern dictionary*, which collects, per country, official and unofficial patterns, such as abbreviations. Continuing our example, the pattern dictionary will contain the fuzzy transformation *corporation* \rightarrow *corp*. The

idea is that the most frequent patterns might affect all business registries and, therefore, the *pattern dictionary* is reusable for all repositories. New patterns can be added by defining a (relative) frequency threshold, a level over which such patterns are widely adopted, or by a human check. In our case, the pattern dictionary has been built via human validation, accepting or rejecting patterns based on common sense.⁵ This system is a one-off exercise and, as the tool is use case-specific, users can decide to enrich the pattern dictionary by running the sub-module over their internal databases, increasing the final matching performances. For instance, through this approach, we discovered that a widely used abbreviation affects the entities that contain the substring *in liquidation*, as it is often truncated as *in liqui*.

For European-based entities, we also collected a dictionary of officially used abbreviations, that we also share. For instance, in Germany, the legal business type *Investmentaktiengesellschaft* is usually abbreviated as *InvAG*, and all German-based entities containing such substring can be replaced by the corresponding abbreviation.

Country-Specific Processing. Depending on the business registry rules, country information might be inserted in one of the internationally recognized different formats, such as the ISO codes or the country name itself. Therefore, an entity-matching tool needs to swiftly recognize each format and move from one system to another in a seamless way. Our tool has been designed to support the following international standards: *ISO alpha-2*, *ISO alpha-3* and *M49 code*, plus the country names version, which includes some heuristics designed to treat specific country names, such as *Republic of* or abbreviations, e.g., the USA or the UK. The type of formats used by the business registries must be specified by the user to the tool, which then will handle them.

Address-Specific Processing. If not collected properly, street addresses are, by nature, messy, with multiple formats allowed and much information being usually omitted or abbreviated. Furthermore, different jurisdictions can have very different address systems, adding complexity to treating such attributes as country-specific processing needs to be applied. While the entity name and the country attribute are well-defined and unambiguous attributes, in terms of their content, the same does not apply to the address attribute. Conceptually, an address can usually be made of multiple components, namely, road name, house number, city, and country.

⁵The pattern dictionary we created is integrated into the tool and available in the same repo as a CSV file

ID	Name	Address	...	ID	Name	Address	...
AX3	Alpha Inc.	London Street, 54	...	5237	Alpha Corp.	London Srt	...
GB5	Beta Limited	Calle Central, 4I	...	8472	Beta Ltd	CLL Central, 4	...
...

ID1	ID2	Score
AX3	5237	0,84
GB5	8472	0,94
...

Figure 2: Mapping table, connecting primary keys, i.e., IDs, of the initial business registries.

First, as for the rest of the character attributes, the standard pre-processing is applied. Then, our tool implements a *standardization strategy* of addresses. Given the high instability of this attribute, we apply a set of heuristics that re-order the content of the address attribute: all integers are moved at the start of the string, followed by characters, with whitespaces deleted. Such standardization might lead, in some cases, to apparently mixing different parts of the string. For instance, let’s suppose we have the street address: “1st July Road, 25, Manchester”, with 25 being the corresponding house number. Our standardization strategy would convert the street address as: 125STJULYROADMANCHESTER, with the house number being mixed with the road name. However, such transformation provides more consistency in the join, as it normalizes all addresses and helps minimize the distance between two strings. Finally, an abbreviation dictionary is applied, and country names are removed from the address. The abbreviation dictionary for the address is also available and has been constructed using the technique outlined in Algorithm 1.

Alternative Address Attributes. In some cases, addresses might be parsed into multiple attributes. In our tool, we support cases where postal codes and cities are stored in dedicated attributes and specific pre-processing methodologies are envisioned. For instance, we use the information in these fields to remove the redundancy in the address attribute.

3.3 Matching Records

Once processed and cleaned, the target business registries, \mathcal{D}_1 and \mathcal{D}_2 can be joined to create a *mapping table* \mathcal{B} , as in Figure 2. The *mapping table* is a three-column table, with the identifiers of the two business registries, i.e., the primary keys, as the first two attributes. The third attribute of the table is a *similarity score*, which is computed by the tool and provides a confidence degree of the pair match.

Linking Business Registries. The join over the *business registries* is performed by applying a set γ of con-

ditions, which the user can control. Such conditions are restricted to a set of minimal requirements that we consider the baseline for obtaining an acceptable level of record-matching precision. In particular, we require that at least the name and country attributes are available between the business registries, which can also be considered an essential requirement for the business registry itself and, therefore, almost always satisfied. In the join, we require the equality of such attributes, and we set the country attribute as the default blocking rule, which creates blocks for records that share the same country and reduces the number of comparisons. In addition, the following conditions can be added:

1. String similarity or equality on the address. We will use the Levehnstein distance as our string similarity algorithm, but any other algorithms can be easily implemented. Users can arbitrarily choose and parameterize the distance level.
2. Equality on postal code information.
3. Equality on cities’ names.
4. Equality on other custom attributes, for which no pre-processing is implemented.

The tool supports any combination of the above conditions. BeRTo doesn’t allow users to choose to apply string similarity over the entities’ legal names, as we empirically observed far too many incorrect results while testing it.

Similarity Score. For each match found, we compute a similarity score, regardless of the conditions set by users. This is computed based on the concept of *string similarity*, i.e., the number of edit operations needed to transform a string into another one (Wandelt et al., 2014). Anyone interacting with the mapping table can use such a score to filter inaccurate matches further, after the mapping table has been built. In other words, our similarity score accounts for possibly harmful fuzzy transformations applied to strings. The score is obtained by a mean over the ratio of differences found per attribute used in the join, considering only basic string pre-processing techniques such as accent conversion. Formally, the similarity score for an entity e is computed as:

$$S_e = \frac{1}{N} \sum_{k=1}^N \frac{\text{Levenshtein}(k_1, k_2)}{\text{len}(k_1) + \text{len}(k_2)} \quad (1)$$

where N is the number of attributes used in the join, while k_1 and k_2 denote the same string attribute in \mathcal{D}_1 and \mathcal{D}_2 for the same entity e , respectively.

Precision-Recall Trade-Off. The available attributes and the join conditions, chosen by users, influence

the results and the position over the precision-recall trade-off. With precision, we indicate the case in which we minimize the number of potential mistakes in the record linkage at the cost of discarding good-but-not-perfect matches, which might include true matches. With recall, we indicate the opposite case, in which likely matches are included in the result. For instance, the latter case might be chosen by users who are performing aggregate and non-critical analysis in which wrong matches contribution is mitigated by the overall accuracy of the system. The maximum precision is reached by requiring equality over all attributes available and by disabling the use of the name dictionaries containing the *fuzzy pattern rules*, discussed in Section 3.2. The maximum recall is reached by running the name-country-only version of the tool. In the middle, all potential combinations of attributes and string similarity thresholds allow us to adapt the tool to more balanced trade-offs.

3.4 System Implementation

BeRTo has been implemented in PySpark (Drabas and Lee, 2017), an open-source Apache Spark Python API that facilitates the development of large-scale, distributed data processing applications and leverages Hadoop as its technological framework. PySpark combines the strengths of the Python programming language with resilient distributed dataset (RDD) abstraction, to enable parallel data processing, making it well-suited for tasks such as record linkage. Hadoop, with its distributed file system (HDFS) and MapReduce programming model, provides a robust infrastructure for storing and processing large volumes of data across clusters of commodity hardware.

4 EXPERIMENTAL EVALUATION

In this section we first present our data sources and our environment settings, then we present our experiments and results, assessing our entity-matching tool. We aim to demonstrate the effectiveness of our tool, in terms of precision and recall, in the entity matching task, by running it over a controlled scenario in which common identifiers are available and we discuss how parameters can be chosen according to the desired positioning in the precision-recall trade-off. Then, we evaluate the scalability of our solution. We will benchmark with two state-of-the-art general-purpose record linkage open-source tools, Dedupe.io and Splink, and show how BeRTo behaves under different configurations.

Environment and Settings. The experiments were conducted on a dedicated server running Ubuntu 18.04.5 LTS with a 48-core Intel Xeon Gold 5118 CPU, 376GB of RAM, and 781 GB of swap partition. The server hosted both Apache Hadoop 3.3.6 and Apache Spark 3.5.0. Hadoop has been configured to run on a single node cluster in pseudo-distributed mode, while Spark has been deployed in cluster mode with its standalone cluster manager. Spark’s configuration included dynamic allocation with 4 executor cores and 20GB of executor memory, and 16GB of driver’s memory.

4.1 Data Sources

To guarantee the complete reproducibility of our results, we will use only public and openly available data sources, which we detail in Table 1.

Table 1: Data sources used in our experiments.

Dataset	Entities	Available Attributes	Source
ECB List	110k	Name, Addr., City, Postcode, Country	European Central Bank
GLEIF	2.5M	Name, Addr., City, Postcode, Country	Financial Stability Board
National Registries	10M	Name, Addr., Country	Individual Countries

The first data source we use is a list of financial institutions based in the European Union, maintained by the European Central Bank (ECB) and publicly available.⁶ It includes monetary financial institutions, investment funds, financial vehicle corporations, payment statistics relevant institutions, insurance corporations and pension funds, for a total number of around 110k entities, identified by the so-called *RIAD* code.⁷

The Global LEI Index (GLEIF) is a global online source for open, standardized and high-quality legal entity reference data.⁸ The Legal Entity Identifier (LEI) is a 20-character, alpha-numeric code based on the ISO 17442 standard developed by the International Organization for Standardization (ISO). It connects to key reference information, enabling clear and unique identification of legal entities participating in financial transactions. It also includes information about the names and addresses of entities. In addition, the LEI number is also available in the data that

⁶ECB Data: <https://bitly.ws/3b3uQ>

⁷Guideline (EU) 2017/2335 of the European Central Bank of 23 November 2017

⁸GLEIF Data: <https://www.gleif.org/en>

the ECB collects, thus enabling a precision and recall analysis of the entity matching between these two data sources. Finally, we collected national business registry data from individual countries’ statistical offices, which make them available open-licensed, namely the UK, some USA states, Romania, Latvia and India, for a total of 10M entities. Here, only names and addresses were available across the data sources. Therefore, it is a big data source to demonstrate only BeRTo scalability, given the absence of a shared identifier with other datasets that would enable precision analysis. All collected data have been stored in parquet.

4.2 Experiments and Results

To assess the precision and recall abilities of BeRTo, we leverage the ECB List and GLEIF datasets, with the LEI being the golden truth. For evaluating scalability and computational performances, we test BeRTo under different settings and configurations that users might choose, as outlined in Section 3.3. Furthermore, we also adopt a resampling approach to increase the size of our datasets, simulating real company datasets such as Orbis’ and OpenCorporates’ ones, and demonstrating the scalability of BeRTo.

4.2.1 Precision and Recall

We compute precision and recall by adapting their definition to the specific problem, as in (Wang et al., 2011). In particular, we consider *true positives* (TP) as the correct pairs of entities that have been linked, by comparing them with our golden truth benchmark, obtained by joining the ECB List with GLEIF via LEI. The official mapping comprises 61k companies, which we can consider as the total number of companies available in both datasets. In other words, it represents the denominator of the recall. Precision is instead computed by dividing the TPs by the total number of tuples linked by the tool, excluding additional matches, i.e., RIAD-LEI pairs that are completely new and can be used to enrich the ECB List dataset. The presence of additional matches, for which we could not say anything about correctness, is a natural consequence of dealing with real data. In fact, while LEI is the official foreign key between the two datasets, LEIs might still be missing in real-world contexts. We run four configurations of BeRTo to demonstrate its abilities over different settings: recall-focus (R), balanced-recall (BR), balanced precision (BP) and precision-focus (P), as discussed in Section 3.3. Additionally, we run Dedupe by manually tuning it for this domain, and Splink, requiring an exact match on cities and countries, and levenshtein similarity on addresses with a blocking

Table 2: Precision and Recall of BeRTo, Dedupe and Splink. Last column indicates additional matches with respect to the golden truth.

Tool	Recall	Precision	Additional Matches
BeRTo-R	0.67	0.94	12k
BeRTo-BR	0.63	0.96	4k
BeRTo-BP	0.48	0.97	3k
BeRTo-P	0.37	0.99	2k
Dedupe	0.66	0.78	18k
Splink	0.32	0.97	4k

rule on names.⁹

The results are presented in Table 2. BeRTo achieves a very high precision in all its configurations, outperforming Dedupe’s precision in all cases and also maximising recall in the case of BeRTo-R configuration. Overall, the recall obtained by all tools is not too high. This is mostly due to the fact that our experiments are conducted over real data, in which attributes might be missing or incorrect, and cleaning efforts in such cases are irrelevant. Additional matches found are higher for Dedupe, but considering its lower precision, it’s questionable whether all of these are true new matches. Regarding BeRTo’s configurations, we assist in wide drops of recall performance when moving to more precision-focused settings. We consider this a natural behavior of any precision-recall trade-off. BeRTo also returns a similarity score for each ID pair found, as described in Section 3.3. The average similarity scores in these experiments have been 0.85 for BeRTo-R, 0.89 for BeRTo-BR, 0.90 for BeRTo-BP and 0.97 BeRTo-P.

Computational Performances. We then tested BeRTo on its scalability performances. To this aim, we first present the average running times (over five runs) for the experiments conducted in the previous section in Figure 3. Execution times have been computed on the actual running time of the tools, i.e., not considering the dataset loading time and, for Dedupe, excluding the required active learning labeling task.

In this experiment, BeRTo is nearly as efficient as Splink in the record linkage task. However, we had to set up Splink with a blocking rule on names whose recall is much lower than BeRTo’s. Under other blocking rules, for instance, by blocking on countries as our tool, we experienced out-of-memory Spark errors, which highlights its much higher computational costs since we conducted the experiment under the same Spark configuration of BeRTo. Dedupe lags far behind, with its execution time being more than eight times higher than both BeRTo’s and Splink’s ones. Although a Spark implementation of Dedupe would

⁹Testing other blocking rule combinations, we experienced worse precision-recall performances

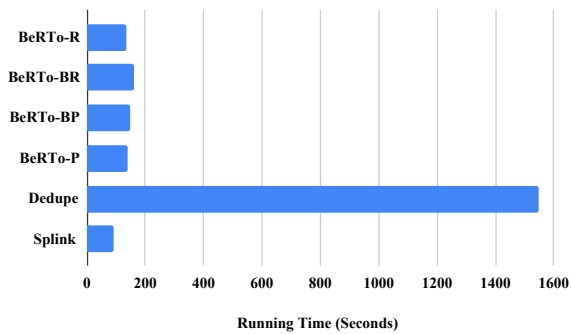


Figure 3: Average running times over five runs in the ECB List-GLEIF record linkage experiment.

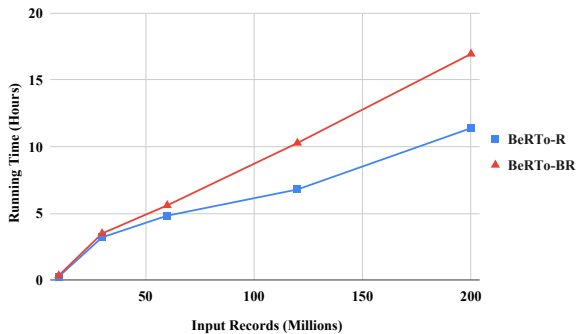


Figure 4: Running time results of merging the GLEIF and National Registries datasets. Input records refer to the total number of legal entities summing the two data sources. We increased the input dataset sizes via resampling with replacement, maintaining the original relative proportions.

surely recover speed, we recall its much lower precision when compared to BeRTo and the need for active learning from users.

Then, we tested the execution times of BeRTo in a big data setting, i.e., linking the GLEIF and National Registries datasets. As mentioned, nothing can be said about precision and recall for this scenario. Therefore, we limit to observe the results in terms of the running times of the tool. As BeRTo-BP and BeRTo-P configurations require the presence of additional address’ attributes, namely cities and postcodes, we can run only BeRTo-R and BeRTo-BR configurations, as the National Registries data source only contains a generic attribute of addresses. In this experiment, we also adopted an incremental resampling approach of the data sources, reaching similar sizes of the largest worldwide data sources of legal entity data, e.g., Orbis and OpenCorporates. Results are presented in Figure 4. Both tested configurations display a sub-linear increase in running time, making the tool is well-suited for industrial deployments in a big-data context, where regular updates and runs are required. As expected, BeRTo-R demonstrates a shorter running time, due to fewer attributes involved

in the join operation. However, we recall the superior precision of BeRTo-BR, which justifies the observed difference in running time.

4.3 Discussion and Limitations

While obtaining an overall good trade-off in the experiments, demonstrating better overall performances compared to state-of-the-art tools we analyzed, we acknowledge that few reliable and open-data repositories of company information, especially the ones containing the golden truth, i.e., shared identities, are available. Therefore, we can’t guarantee the same level of accuracy across all business registry repositories. In addition, we recognize that our solution might be European-biased since it has been mostly trained on data from European countries. However, BeRTo’s open-source nature allows the community to contribute to optimization, for instance, by using the fuzzy system to enrich the dictionary of patterns with repository-specific characteristics.

Regarding our benchmarks, we report how both Dedupe and Splink require a significant effort to be set up, either for performing active learning, such as in the case of Dedupe, or to find the complex set of setting combinations to adapt the tool to the problem at hand, a particularly relevant and time-consuming activity. BeRTo, instead, offers a pre-defined and flexible set of possible configurations and choices, achieving a more user-friendly setup. While it is not general-purpose as Dedupe or Splink, we think that similar solutions can be easily implemented by replicating the same components of our tool in other recurring and relevant record linkage activities, such as linking persons or financial transactions, to obtain domain-aware tools that are user-friendly and less time consuming, with users only tasked to choose their preferred recall-precision configuration.

5 CONCLUSIONS

We presented BeRTo, an open-source end-to-end record linkage tool that exhibits high precision and scalability. By being domain-specific, our tool can integrate pre-processing, data cleaning components, and fuzzy-based rules that allow it to achieve similar or higher precision-recall results to general-purpose and machine learning-based record linkage tools, but with the advantages of being user-friendly and scalable, thus fully adaptable to big data and fast-paced environments, in which record linkage activities are performed daily. Furthermore, it doesn’t require a golden truth to be tuned on, but, if available, can be

used as an additional resource to improve the final results. We have also developed a plain Python version of BeRTo for smaller record linkage tasks, which we have also made available in our repository. While not scalable, it can be used for small data business registry record linkage. For future work, we will work on a graphical user interface for BeRTo, unlocking even more user-friendly uses of our tool.

ACKNOWLEDGEMENTS

Andrea Colombo kindly acknowledges INPS for funding his Ph.D. program.

REFERENCES

- Arief, R., Achmad, B. M., Tubagus, M. K., and Hustinawaty (2018). Automated extraction of large scale scanned document images using google vision ocr in apache hadoop environment. *International Journal of Advanced Computer Science and Applications*, 9(11).
- Bajgar, M., Berlingieri, G., Calligaris, S., Criscuolo, C., and Timmis, J. (2020). Coverage and representativeness of orbis data. *OECD Library*.
- Bilenko, M., Kamath, B., and Mooney, R. J. (2006). Adaptive blocking: Learning to scale up record linkage. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 87–96. IEEE.
- Chen, X., Schallehn, E., and Saake, G. (2018). Cloud-scale entity resolution: current state and open challenges. *Open Journal of Big Data (OJBD)*, 4(1):30–51.
- Christen, P. (2012). *The data matching process*. Springer.
- Christophides, V. e. a. (2020). An overview of end-to-end entity resolution for big data. *ACM Comput. Surv.*, 53(6).
- Dong, X. L. and Srivastava, D. (2013). Big data integration. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 1245–1248.
- Drabas, T. and Lee, D. (2017). *Learning PySpark*. Packt Publishing Ltd.
- Ebraheem, M., Thirumuruganathan, S., Joty, S., Ouzzani, M., and Tang, N. (2018). Distributed representations of tuples for entity resolution. *Proc. VLDB Endow.*, 11(11):1454–1467.
- Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. (2007). Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16.
- Eurostat (2024). Statistical business registers. <https://ec.europa.eu/eurostat/web/statistical-business-registers/information-data> Accessed: 01/2024.
- Forest, G. and Eder, D. (2015). Dedupe. <https://github.com/dedupeio/dedupe>. [Accessed January-2024].
- Gagliardelli, L., Simonini, G., Beneventano, D., Bergamaschi, S., et al. (2019). Sparker: Scaling entity resolution in spark. In *Advances in Database Technology-EDBT 2019, 22nd International Conference on Extending Database Technology, Lisbon, Portugal, March 26-29, Proceedings*. PRT.
- Getoor, L. and Machanavajjhala, A. (2012). Entity resolution: Theory, practice & open challenges. *Proc. VLDB Endow.*, 5(12).
- Guralnick, R. P. e. a. (2015). Community next steps for making globally unique identifiers work for biocollections data. *ZooKeys*, (494):133.
- Herzog, T. N., Scheuren, F. J., and Winkler, W. E. (2007). *Data Quality and Record Linkage Techniques*. Springer Publishing, 1st edition.
- Kolb, L., Thor, A., and Rahm, E. (2012a). Dedoop: Efficient deduplication with hadoop. *Proc. VLDB Endow.*, 5(12):1878–1881.
- Kolb, L., Thor, A., and Rahm, E. (2012b). Load balancing for mapreduce-based entity resolution. In *2012 IEEE 28th International Conference on Data Engineering*, pages 618–629.
- Konda, P., Das, S., C., P. S. G., Doan, A., Ardalan, A., Ballard, J. R., Li, H., Panahi, F., Zhang, H., Naughton, J., Prasad, S., Krishnan, G., Deep, R., and Raghavendra, V. (2016). Magellan: Toward building entity matching management systems over data science stacks. *Proc. VLDB Endow.*, 9(13):1581–1584.
- Koudas, N., Sarawagi, S., and Srivastava, D. (2006). Record linkage: Similarity measures and algorithms. In *Proceedings of the 2006 ACM SIGMOD International Conference, SIGMOD '06*, page 802–803. Association for Computing Machinery.
- Krishnan, S., Haas, D., Franklin, M. J., and Wu, E. (2016). Towards reliable interactive data cleaning: a user survey and recommendations. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics, HILDA '16*. ACM.
- Li, Y., Li, J., Suhara, Y., Doan, A., and Tan, W.-C. (2020). Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment*, 14.
- Linacre, R., Lindsay, S., Manassis, T., Slade, Z., Hepworth, T., Kennedy, R., and Bond, A. (2022). Splink: Free software for probabilistic record linkage at scale. *International Journal of Population Data Science*, 7(3).
- Nentwig, M. and Rahm, E. (2018). Incremental clustering on linked data. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 531–538. IEEE.
- Ryan, L., Thompson, C., and Jones, J. (2020). A statistical business register spine as a new approach to support data integration and firm-level data linking: An abs perspective. *Statistical Journal of the IAOS*, 36(3):767–774.
- Salloum, S., Dautov, R., Chen, X., Peng, P. X., and Huang, J. Z. (2016). Big data analytics on apache spark. *International Journal of Data Science and Analytics*, 1:145–164.
- Shaikh, E., Mohiuddin, I., Alufaisan, Y., and Nahvi, I. (2019). Apache spark: A big data processing en-

- gine. In *2019 2nd IEEE Middle East and North Africa COMMUNICATIONS Conference (MENACOMM)*.
- Wandelt, S., Deng, D., Gerdjikov, S., Mishra, S., Mitankin, P., Patil, M., Siragusa, E., Tiskin, A., Wang, W., Wang, J., and Leser, U. (2014). State-of-the-art in string similarity search and join. *SIGMOD Rec.*, 43(1):64–76.
- Wang, J., Li, G., Yu, J. X., and Feng, J. (2011). Entity matching: how similar is similar. *Proc. VLDB Endow.*, 4(10):622–633.
- Wang, J., Li, Y., and Hirota, W. (2021). Machamp: A generalized entity matching benchmark. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management, CIKM '21*, page 4633–4642, New York, NY, USA. Association for Computing Machinery.
- Yan, Y., Meyles, S., Haghghi, A., and Suciu, D. (2020). Entity matching in the wild: A consistent and versatile framework to unify data in industrial applications. *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*.