Machine Learning and Deep Learning approaches for Record Linkage with unstructured data

Mattia Micaloni, Carlo Proserpio, Alessandro Pesare

Keywords: Big Data Integration, Large Language Model (LLM), Record Linkage, Entity resolution, Embedding, Blocking, Pairwise matching, Clustering, Siamese Network

1. Introduction

In Big Data analysis, where data are heterogeneous, Data Integration is useful to synthesize insights from different sources. In the context of Data Integration in Big Data, in addition to the variety of sources and of data, it is important to consider the large volume of data and the velocity with which it might change, besides the fact that not all data have the same value, they may have low or no information content. This data often comes in various formats, including structured data (e.g., relational databases), semi-structured data (e.g., XML, JSON), and unstructured data (e.g., text, images, videos). Additionally, the velocity at which data is generated necessitates real-time or near-real-time processing capabilities to derive actionable insights.

The integration process typically consists of three parts [Len02]: Schema Alignment, Record Linkage and Data Fusion. Each part usually comprises three steps. Schema Alignment is the initial part in data integration, it involves integrating and reconciling different data schemas to ensure data interoperability and consistency across diverse systems [Zoh11]. Record linkage, also known as data matching or entity resolution, is the process of identifying and merging records that refer to the same entity across different data sources [EIV07]. It involves several steps to efficiently and accurately match records, which include blocking, pairwise matching, and entity clustering. Blocking is a technique used to reduce the number of comparisons needed for record linkage. Instead of comparing each record with every other record, which can be computationally expensive, records are grouped into smaller subsets, or "blocks", based on certain key attributes. Comparisons are then made only within these blocks. Pairwise matching involves comparing pairs of records within each block to determine if they refer to the same entity. This step uses various similarity measures and matching algorithms to assess the likelihood that two records are a match. Clustering implies the grouping of all records that are believed to refer to the same entity based on pairwise matching results. Data fusion is the last part

of the process of integrating the sources to produce more consistent, accurate, and useful information. It addresses the challenge of combining diverse data while resolving conflicts and ensuring data quality [DBS09].
The objective of this paper is to investigate the challenges posed by Record Linkage in the context of Big Data and propose a solution to address them. In addition, we tried the usage of Large Language Models (LLMs) and Siamese Networks for pairwise matching. Our proposed solution will be tested on a dataset from the DI2KG challenge.

2. Problem Definition

In the ever-expanding field of data management, one of the most critical and complex challenges is the accurate integration of datasets from multiple sources. This process, known as data integration, is essential for enabling comprehensive data analysis and decision-making. A fundamental step within data integration is record linkage (also known as entity resolution or entity matching), which aims to identify and merge records that refer to the same real-world entity despite possible differences in data format, structure, or quality. The accuracy of record linkage directly impacts the quality of data integration and, consequently, the reliability of insights derived from the data.

2.1 The Complexity of Big Data

Big Data is characterized by five dimensions, known as the 5Vs [KYK20]: volume, variety, velocity, veracity, and value. Volume refers to the sheer amount of data generated continuously, which can be overwhelming in scope. Variety indicates the diversity of data types and sources, including structured databases, semi-structured files like XML, and unstructured formats such as images and text. Velocity describes the rapid rate at which data is produced and must be processed. Veracity pertains to the trustworthiness and accuracy of the data, and Value represents the potential benefits and insights that can be extracted through analysis. These dimensions illustrate the inherent complexity of managing Big Data, particularly in tasks such as record linkage, where the goal is to unify diverse and dynamic data sources into a coherent and accurate dataset. Each dimension introduces specific challenges that must be addressed to ensure effective data integration.

2.2 Big Data Record Linkage

Record linkage in the context of Big Data presents several unique challenges that differentiate it from traditional data integration efforts. Traditional record linkage methods often depend on measuring the similarity between a static set of structured tuples that share the same schema. However, Big Data introduces a new level of complexity due to its volume, variety, and velocity. The sheer number of records that need to be processed can be staggering, making traditional methods inefficient. For instance, comparing each pair

of records in a dataset with millions or billions of entries is computationally expensive and time-consuming. Furthermore, Big Data involves integrating data from numerous sources, each with potentially different formats and structures. This heterogeneity makes it difficult to apply uniform rules for record linkage. Structured data may need to be linked with unstructured text or semi-structured files, requiring advanced techniques to identify and merge corresponding records accurately.
The dynamic nature of data sources in Big Data means that data is continuously evolving. Traditional record linkage methods, which often involve processing the entire dataset from scratch, are impractical in such environments. Incremental record linkage methods are necessary to update the linkage process efficiently as new data is added or existing data is modified. Moreover, typographical errors, inconsistent field entries, and varying data representations further complicate the linkage process. Selecting the appropriate fields for similarity measurement and determining the threshold for considering records as duplicates are non-trivial tasks that require careful consideration and fine-tuning, we will see examples of that in our implementation.

2.3 The importance of Record Linkage

Ensuring high-quality data integration through effective record linkage is crucial for any data-driven analysis. Data preparation and cleaning, including record linkage, are essential steps in the analytical process, often consuming a significant portion of the total time and resources invested in data projects. For example, it is estimated that up to 80% of the time spent on data analysis is devoted to preparing and cleaning datasets [SI+18].
Without accurate record linkage, the resulting dataset may contain duplicate records or fail to merge all relevant data, leading to incomplete or misleading analysis. Thus, overcoming the challenges of record linkage in Big Data is vital for extracting reliable and actionable insights from integrated data sources.

3. Proposed solution

Our approach consist in solving the problem of record linkage using deep learning techniques and Large Language Models (LLMs).
[AD23] and [Li+24] already propose different methods for enhancing Record Linkage with the help of LLMs, but leave room for implementation choices and number of steps involved in the process of Record Linkage.
Our methodology can be summarized into four main activities: dataset preprocessing, embedding data items into a vector space, clustering for the blocking step and pairwise matching to find related entities. This combination aims to address the inherent challenges of record linkage, particularly its scalability and accuracy.
The preprocessing stage is crucial in identifying the most representative features of each data item in the dataset. Effective preprocessing and data cleaning are fundamental to

achieving good results when employing machine learning models. In our case, preprocessing focuses on extracting and refining the information that best represents each data item. This step ensures that the subsequent embedding and clustering processes work with clean, relevant data, thereby improving the overall performance of the record linkage system.

Once the dataset is preprocessed, the next step involves using an embedding model to map data items into a vector space. The goal here is to represent similar objects as similar vectors. This step is critical because it transforms the data into a format suitable for machine learning models to process and analyze. By using a language model for embedding, we can capture the semantic similarities between data items, which is essential for accurately identifying potential matches.

Clustering is used to implement the blocking step, which is essential for the scalability of the proposed system. Record linkage, in its worst-case scenario, involves an exponential complexity due to the need to compare all possible pairs of records. Blocking mitigates this by grouping similar data items into clusters, thereby reducing the number of comparisons needed. The effectiveness of the blocking step hinges on ensuring that similar data items, which are candidates for being the same real-world entity, end up in the same cluster.

The final step is pairwise matching, where we compare pairs of records within each block to verify if they represent the same entity. This stage leverages LLMs or Siamese Networks to perform the comparisons, using the context provided by the discriminant field used for embedding (e.g., page titles). Initially, most of the information in the discriminant field is reduced during preprocessing to standardize the data. However, when using an LLM for pairwise matching, having more contextual information is beneficial. Therefore, we retain the raw discriminant field data before pre-processing to provide the LLM with richer context during matching.

In Figure 1 we summarized the process of our approach. The effectiveness of our approach is evaluated through detailed experiments. We assess the performance of the blocking strategies and the pairwise matching phase using precision, recall, and F-measure metrics. These evaluations help determine the accuracy and efficiency of our methodology in handling the record linkage problem. Our approach aims to demonstrate the viability of using deep learning and LLMs for record linkage. By integrating these advanced techniques, we seek to improve the scalability and accuracy of record linkage systems, addressing the challenges posed by Big Data integration.

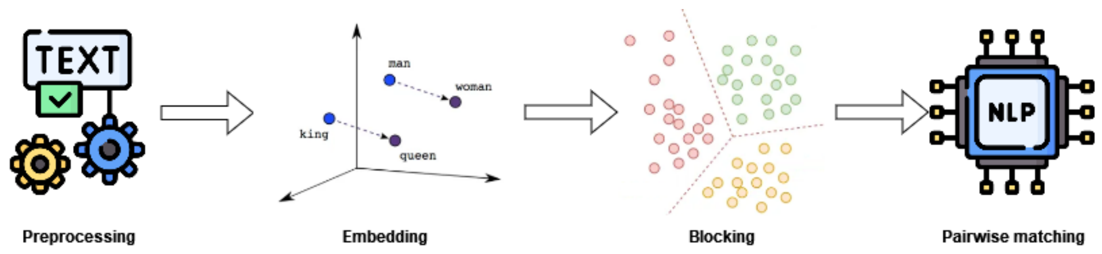

Figure 1: Our process for Record Linkage in Big Data Integration.

4. DI2KG Dataset

The dataset used is the Monitor Specifications from the challenge DI2KG[1]. It has 26 sources with 16662 products, it exhibits high degree of heterogeneity both across and within sources. This dataset consist of a set of products specifications in JSON format, automatically extracted from multiple e-commerce websites. Each specification has been extracted from a web page and refers to a real-world product. A specification consists of a list of pairs and is stored in a file. Files are organized into directories, each directory corresponds to a web source.
An example of specification (for the source www.ebay.com) could be:

```
{
    "<page title>": "ASUS VT229H | ebay.com",
    "screen size": "21.5''",
    "brand": "Asus",
    "display type": "LED",
    "dimension": "Dimensions: 19.40 x 8.00 x 11.80 inches",
    "refresh rate": "75hz",
    "general features": "Black",
}
```

5. Our implementation

5.1 Preprocessing

Since the most informative content for identifying products is given by the model, our goal is to try to identify as many strings as possible that would represent it. Some of the sources, among the JSON attributes, already included the model, so whenever possible we tried to derive the model directly from the attributes. We firstly identified the attributes that contains the model. If an attribute containing only the model is present we selected it, otherwise we prioritize attributes that contains few more information in

[1]http://di2kg.inf.uniroma3.it/datasets.html

addition to the model to easily extract the model. When none of the relevant attributes were found we selected the page title which is the only common attribute among all the sources. After retrieving the relevant text, we cleaned the data: the text was put all in lowercase; with regular expressions we identified and removed websites and substrings that were not necessary for product identification such as units of measurement (inches, weights, dates, Hertz), display resolutions, and uninformative frequent words (e.g. monitor, led, lcd, display); we removed spaces between adjacent numbers since some times for readability the patterns of the models were not written as a single word; finally, we removed extra whitespace and wrote the preprocessing result in a single JSON in the format *"<source>//<item>" : "<cleaned text>"*, where source is the link of the source and item is the name of the JSON containing the data for that item.

5.2 Embedding

In our approach, we employ DistilBERT, a distilled version of the BERT (Bidirectional Encoder Representations from Transformers) model, for embedding the data items into a vector space. DistilBERT offers a significant reduction in size and computational overhead while retaining a substantial portion of BERT's performance capabilities [San+19]. This makes it an ideal choice for embedding in large-scale record linkage tasks. The use of DistilBERT allows us to capture the semantic similarities between data items efficiently. By converting textual data into dense vector representations, DistilBERT ensures that similar items are represented by closely positioned vectors in the embedding space. This semantic embeddings is crucial for the subsequent clustering and pairwise matching steps, as it provides a robust foundation for identifying potential matches with high accuracy. The model's ability to understand context and nuances in language enhances the quality of the embeddings, leading to more effective blocking and matching processes in the record linkage pipeline. In our case DistilBERT transformed each preprocessed string into a 768-dimensional vector.

5.3 Blocking

Based on the embeddings, we realized blocking with different clustering algorithms.
The algorithms we used are:

Agglomerative : This is a bottom-up approach for hierarchical clustering, each data point starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy, based on some distance metric. [scia]

DBSCAN : (Density-Based Spatial Clustering of Applications with Noise) It is a density-based approach for clustering. Given a parameter ϵ, the algorithm denotes as *core-points* the points that have at least a certain number of neighbours at distance ϵ and marks as outliers the points that lie alone or with few neighbours. It can identify clusters with complex shapes and thus it is preferred to classical clustering algorithms. [scib]

HDBSCAN : (Hierarchical Density-Based Spatial Clustering of Applications with Noise) It performs DBSCAN over varying ϵ values and analyzes the result to find a clustering that gives the best stability over ϵ. The stability measures how persistent a cluster is across different distance thresholds (varying ϵ values). A cluster is persistent in the sense that it is still considered as one cluster rather than two separate clusters. This allows HDBSCAN to find clusters of varying densities and be more robust to parameter selection. [CMS13]

K-means : this is a method of clustering that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean distance to a center point called *cluster centroid*. The centroids can be chosen randomly or with a probability which is proportionally inverse to the distance between a centroid candidate and all other already chosen centroids. [scic]

5.4 Pairwise matching

The final step of the Record Linkage pipeline is the Pairwise matching, which is the process of comparing the entities in pairs, to understand if they represent the same object. In the context of Big Data and this project, it is clearly inefficient to address this step doing a naive pairwise control between all entities in the dataset. Hence, in our project we used a blocking approach as described in 5.3 to reduce the number of pairs to analyze. We evaluated the similarity between all entities of the same block. The number of combinations decreases drastically from 138.802.791 to just 79.411 using the 1267 clusters (blocks) we calculated in the previous step. To do this last step, we used both an LLM-based approach and a Siamese Networks approach.

5.4.1 Pairwise matching using LLMs

The idea behind the approach using Large Language Models is simply to engineer a prompt (example in Figure 2) so to have the LLM decide which pair of entities are the same object (in this case the same monitor) with a certain precision. This method heavily relies on the complexity of the LLM itself. All the models were tried inside a Google Colab environment using the TPU (Tensor Processing Unit) whenever possible and were loaded in GGUF file format using the *llama-cpp-python* python library. We tried models with few billions parameters such as Mistral 7B[2] or Mistral 22B[3], which yielded delusional results, or even did not understand the question correctly. After these results we moved to higher parameter count models such as LLama3 70B[4] and ChatGPT-4o through the

[2]https://huggingface.co/TheBloke/Mistral-7B-Instruct-v0.2-GGUF
[3]https://huggingface.co/bartowski/Mistral-22B-v0.2-GGUF
[4]https://huggingface.co/bartowski/L3-70B-Euryale-v2.1-GGUF

official website. The LLama3 model performed really well but had inference times of over 30 seconds using the Google Colab TPU. Using any other type of hardware provided by Google Colab free account such as CPU or T4 GPU would never produce a response. On the other hand, not surprisingly, ChatGPT-4o gave good results with minimal inference times, although this comparison is not fair hardware-wise. Given that the number of queries to send are exactly the number of combinations, using blocking lets this approach be somewhat scalable. For reference, we have 79411 combinations and the average number of tokens per query is approximately 150 (depending if the model has run out of the initial context or not). So if we wanted to use the OpenAI API to query ChatGPT-4o for example, we would pay $5,00$ USD for a million input tokens, and in our case we have more or less 12 million tokens in input. Hence, it would be feasible to scale money-wise, especially if some other less powerful models are used, such as ChatGPT-3.5 Turbo which is priced at $0,50$ USD per million input tokens.
All things considered, this approach works pretty well and enables us to identify objects among duplicated entities even if they differ in structure or syntax. It is important to note that it could work better depending on the training data of the model (in the case of LLMs, strings).

```
def query_llama(title1, title2, begin: bool):
    prompt = ""
    if begin:
      prompt += "<|begin_of_text|>"
    prompt += f'''<|start_header_id|>system<|end_header_id|>
    You are an helful assistant that can tell if two monitors are the same object just by analyzing their product webpage title.
    To help yourself search for model names or alphanumerical strings and try to ignore the webpage name in the webpage titles.
    If the page titles represent the same entity your answer MUST BE "yes".
    If the page titles do not represent the same entity your answer MUST BE "no".

    Example 1:
    first page title: "Hp Hewlett Packard HP Z22i D7Q14AT ABB Planet Computer.it"
    second page title: "HP Z22i - MrHighTech Shop"
    "yes"
    Example 2:
    first page title: "Hp Hewlett Packard HP Z22i D7Q14AT ABB Planet Computer.it"
    second page title: "C4D33AA#ABA - Hp Pavilion 20xi Ips Led Backlit Monitor - PC-Canada.com"
    "no"
    <|eot_id|><|start_header_id|>user<|end_header_id|>
    Now tell me if these two webpage titles represent the same object:
```

```
    first webpage title: "{title1}"
    second webpage title: "{title2}"
    Answers MUST BE "yes" or "no"
    <|eot_id|><|start_header_id|>assistant<|end_header_id|>'''

    output = llm(prompt, max_tokens=30, temperature=0.15, echo=False )
    response = output['choices'][0]['text']

    return response
```

Figure 2: Example of prompt for Llama3

5.4.2 Pairwise matching using Siamese Network

Our last approach was based on Siamese networks, a Siamese Network is an artificial neural network composed by two Multilayer Perceptron (MLP) with the same parameters and architecture, as shown in figure 3.

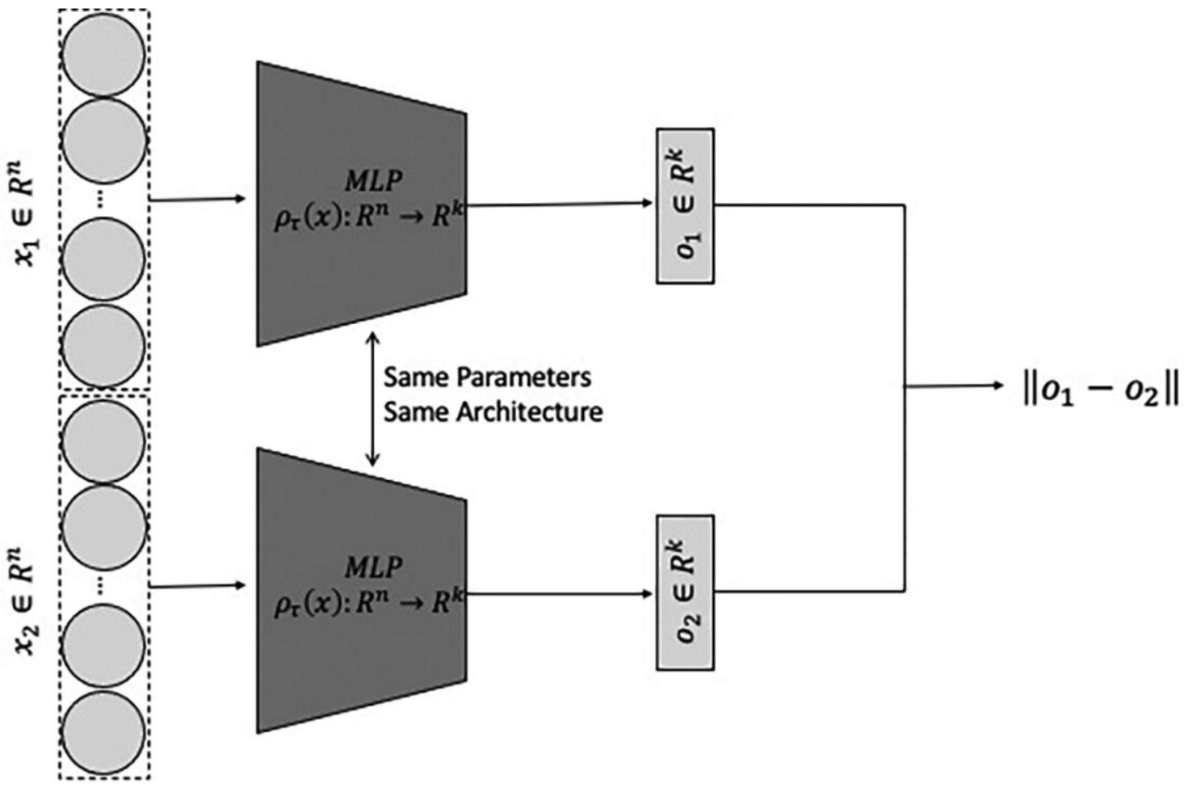

Figure 3: The architecture of a Siamese Network.

In 2020 [Jur21] proposed a novel approach to linking unstructured data based on the application of the Siamese Multilayer Perceptron model using a Contrastive Loss:

$$L = y \cdot ||o_1 - o_2||^2 + (1 - y) \cdot max(margin - ||o_1 - o_2||,\ 0)^2$$

where $o_1, o_2 \in \mathbb{R}^k$ are the network outputs obtained from the embedding vectors $x_1, x_2 \in \mathbb{R}^n$ as input to the MLPs $\rho_\tau(x) : \mathbb{R}^n \longrightarrow \mathbb{R}^k$. We have that if $y = 0$ the embedding vectors x_1 and x_2 do not represent the same entity and $y = 1$ otherwise. The *margin* parameter plays a role in tightening the constraint during the learning process. When two records do not correspond to the same entity (i.e., they do not match), the distance between them must be at least equal to the margin. In the classification phase, a new pair of embedded vectors is fed into the model, which generates their updated representations. The Euclidean distance between these output vectors is then computed. Using this distance, along with a predefined threshold, the pair is classified as either a match or a non-match.

In the process of training the model, we generated a training set and a test set (80/20 split) from the ground truth file that comes alongside the dataset (explained better in section 6). We trained the model for a total of 20 epochs, as the contrastive loss converged quickly. For each training epoch, we fed the model with only a portion of non-matching pairs to avoid creating an overly strong bias towards predicting non-matching pairs over matching pairs. However, a bias of that kind in the context of record linkage is in fact useful since the non-matching pairs usually overcome easily the matching pairs in terms of quantity. For this reason, we still trained the model on a no-match/match ratio equal to 2.

$$\frac{\#\text{non-matching-pairs}}{\#\text{matching-pairs}} \tag{no-match/match ratio}$$

6. Evaluation

For the purpose of evaluating our results, we decided to use metrics such as Recall, Precision and F-Measure. We compared our results with a document available from the DI2KG challenge containing a list of items and their relative entity. This document (ground truth) is provided in CSV format containing two columns "spec_id" and "entity_id":

spec_id : is a global identifier for a specification and consists of a relative path of the spec file. Note that instead of "/" the spec_id uses a special character "//" and that there is no extension. For instance, the spec_id "www.ebay.com//1000" refers to the 1000.json file inside the www.ebay.com directory.

entity_id : is an identifier of a real-world product (a progressive identifier without any semantics): two specifications assigned to the same entity_id refer to the same real-world product.

```
entity_id, spec_id
ENTITY#001, www.ebay.com//2
ENTITY#001, catalog.com//1
ENTITY#002, ca.pcpartpicker.com//1
```

Example of ground truth elements

Using this document as a reference we managed to calculate the metrics described above for different blocking techniques and pairwise matching.

6.1 Blocking

In the context of blocking we define the metrics as follows. For each entity we select the largest cluster (block) for that entity. True positives are items that are in the largest cluster and are correctly associated with that entity, false positives are items in the cluster that are not related to the entity and false negatives are items that should be in the cluster but aren't.
We indicate true positives with TP, false positives with FP and false negatives with FN. We can then define the equations for *precision*, *recall* and *f-measure* as:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$f\text{-}measure = \frac{2 \cdot precision \cdot recall}{precision + recall} = \frac{2 \cdot TP}{2TP + FP + FN}$$

Using these metrics we evaluated each clustering algorithm from section 5.3. For the Agglomerative clustering algorithm we got the highest Recall ($\sim$ 0.927) but also the lowest Precision value. With the K-means algorithm we got similar result with a Recall of $\sim$ 0.897 and slightly greater value for Precision. DBSCAN is the most balanced algorithm with a value of $\sim$ 0.634 for the Precision and $\sim$ 0.673 for the Recall. The best algorithm is terms of performance based on the F-measure is the HDBSCAN ($\sim$ 0.677), with a lower Precision compared to DBSCAN but higher Recall. The results are shown in Figure 4.

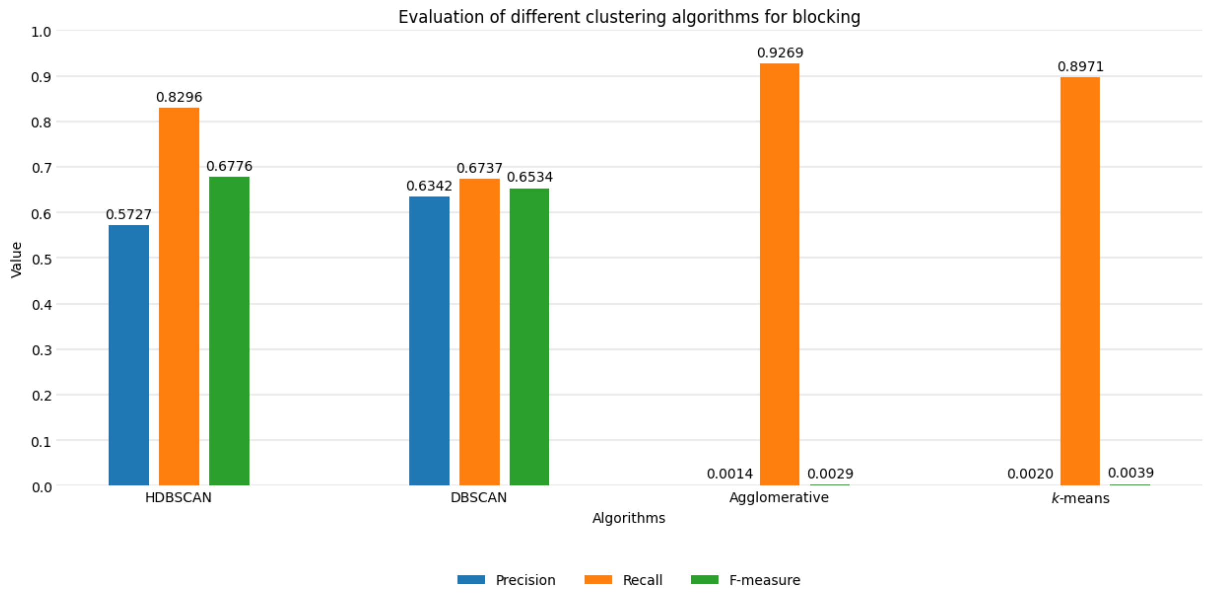

Figure 4: Bar plot for blocking evaluation.

6.2 Pairwise matching using LLMs

Evaluating the pairwise matching step was not trivial. The problem was that computationally we could not run the models in local, so we used Google Colab TPU. As explained in section 5.4, using a free Colab account gave us at maximum an hour and a half of computation. Thus, we managed to test the LLama3 model only for one cluster (cluster 397), which contained 7 elements related to one entity and 2 related to two other entities. The model was tested using the prompt shown in Figure 2 and gave impressive results. It achieved a precision of 0.7 with total recall, giving an f-measure of ~ 0.8235. While these results are not so relevant given the number of clusters tested, it still shows the potential of using LLMs in Record Linkage.
On the same line, ChatGPT-4o shows even more accurate results, at least in the few clusters tested. Using an API to better automate the process could help for better evaluating this model.

6.3 Pairwise matching using Siamese Network

Unlike the LLM evaluation, in the case of Siamese Networks we were able to run them in local with the help of the *scikit-learn* python library with relative ease. This enabled us to run a more detailed evaluation of these models.

Following the training process described in section 5.4.2, we found the best results using a margin of 0.1 and a similarity threshold of $3 \cdot 10^{-5}$. The evaluation was conducted on a test set built from the ground truth file explained in section 6. The model was tested on every match-pair of the test set and a portion of the no-match-pairs inside the test set.

This ratio can be changed arbitrarily to test different no-match/match ratio values.

Siamese Network evaluation	
Precision	99,11%
Recall	69,72%
F-Measure	81,86%

Table 1: Results for the Siamese Network on test set with 29832 pairs, 2712 match-pairs, 27120 no-match-pairs, no-match/match ratio: 10.0

Results for Siamese Network are shown in Table 1. We can deduct from the Precision value that when the model predicts a match, it almost never fails. However, from the Recall value we can see that it doesn't always spot the matches, as some of them are incorrectly predicted as non-matches. The cause is probably that, during the training process, some pairs are not brought close enough in the embedding space by the loss function when they are a match, thus they do not cross the threshold during evaluation. To really strengthen this model, one could try to improve the training process, using more varied data, especially more match-pairs examples.

We also tried to evaluate some clusters to justify the use of this model in the context of Big Data. Unfortunately, most of the elements in the clusters, obtained as explained in section 5.3, are not contained in the ground truth. It is then not feasible to test them, as we do not know exactly if the pairs are a match or not. However, we checked manually by looking through the dataset for the pairs predicted by the model. We do not have precise results but the model seemed to reflect the values obtained in Table 1 or similar. In the most positive scenario, the inter-cluster Recall is limited by the Blocking Recall in Figure 4, but the intra-cluster Recall is much higher.

7. Conclusions

The ability to match records that refer to the same real-world entity is fundamental to producing a unified and coherent dataset, which is crucial for deriving reliable insights and making informed decisions. Without effective record linkage, the integration process would result in fragmented and potentially misleading data, undermining the value of any subsequent analysis. The importance of record linkage is magnified in the context of Big Data, where the volume, variety, and velocity of data introduce unique and significant challenges. Traditional methods, while effective in smaller or less complex environments, often fall short when applied to the vast, dynamic, and heterogeneous datasets characteristic of Big Data. The complexity of integrating structured, semi-structured, and unstructured data from a multitude of sources necessitates advanced techniques that can handle the intricacies of Big Data.

A multitude of approaches to record linkage have been developed, each with its strengths and limitations. Techniques such as blocking, pairwise matching, and clustering are fundamental to traditional methods, but they require substantial computational resources and are often insufficient for the scale and complexity of Big Data. In recent years, the advent of Large Language Models (LLMs) has opened new avenues for record linkage. LLMs, with their ability to understand and generate human-like text, offer promising capabilities for improving the accuracy of pairwise matching, especially in contexts where traditional similarity measures may struggle. By leveraging the contextual understanding of LLMs, it is possible to enhance the identification of matches across diverse and unstructured data sources. However, the application of LLMs in record linkage introduces additional challenges. Scalability remains a significant concern, as the computational resources required to train and deploy LLMs are substantial. Ensuring that LLMs can operate efficiently on the scale required by Big Data is an ongoing challenge that requires careful consideration and optimization. Additionally, LLMs are prone to hallucinations, where the model generates incorrect or nonsensical outputs. This issue can undermine the reliability of record linkage, leading to erroneous matches and potentially compromising the quality of the integrated dataset. On the other hand, using Siamese Networks for record linkage is less computationally intense and still delivers high precision, given good quality of training data. Furthermore, it is far more easy to tweak Siamese Network hyper-parameters to adapt to more precise context or data. In conclusion, record linkage is essential for effective data integration, particularly in the context of Big Data. The vast array of approaches developed to solve this problem highlights its complexity and importance. While traditional methods provide a foundation, the integration of advanced techniques, including LLMs and Siamese Networks, offers promising improvements. However, the challenges associated with scalability and model reliability must be addressed to fully harness the potential of LLMs in record linkage. Ongoing research and innovation in this field will be crucial for advancing data integration methodologies and unlocking the full value of Big Data.

References

[AD23] Abhishek Arora & Melissa Dell. *LinkTransformer: A Unified Package for Record Linkage with Transformer Language Models*. 2023. arXiv: 2309.00789 [cs.CL].

[CMS13] Ricardo J. G. B. Campello, Davoud Moulavi & Joerg Sander. 'Density-Based Clustering Based on Hierarchical Density Estimates'. In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Jian Pei et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 160–172. ISBN: 978-3-642-37456-2.

[DBS09] Xin Luna Dong, Laure Berti-Equille & Divesh Srivastava. 'Integrating conflicting data: the role of source dependence'. In: *Proc. VLDB Endow.* 2.1 (Aug. 2009), pp. 550–561. ISSN: 2150-8097. DOI: 10.14778/1687627.1687690. URL: https://doi.org/10.14778/1687627.1687690.

[EIV07] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis & Vassilios S. Verykios. 'Duplicate Record Detection: A Survey'. In: *IEEE Transactions on Knowledge and Data Engineering* 19.1 (2007), pp. 1–16. DOI: 10.1109/TKDE.2007.250581.

[Jur21] Anna Jurek-Loughrey. 'Deep learning based approach to unstructured record linkage'. In: *International Journal of Web Information Systems* 17.6 (Oct. 2021), pp. 607–621. ISSN: 1744-0084. DOI: 10.1108/ijwis-05-2021-0058. URL: http://dx.doi.org/10.1108/IJWIS-05-2021-0058.

[KYK20] Vinaya Keskar, Jyoti Yadav & Ajay H Kumar. '5V's of Big Data Attributes and their Relevance and Importance across Domains'. In: *International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN* (2020), pp. 2278–3075.

[Len02] Maurizio Lenzerini. 'Data integration: a theoretical perspective'. In: *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. PODS '02. Madison, Wisconsin: Association for Computing Machinery, 2002, pp. 233–246. ISBN: 1581135076. DOI: 10.1145/543613.543644. URL: https://doi.org/10.1145/543613.543644.

[Li+24] Huahang Li et al. *On Leveraging Large Language Models for Enhancing Entity Resolution*. 2024. arXiv: 2401.03426 [cs.CL].

[San+19] Victor Sanh et al. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2019. DOI: 10.48550/ARXIV.1910.01108. URL: https://arxiv.org/abs/1910.01108.

[scia] scikit-learn. *Scikit Learn Agglomerative clustering*. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html [Accessed: 27-06-2024].

[scib] scikit-learn. *Scikit Learn DBSCAN*. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html [Accessed: 27-06-2024].

[scic] scikit-learn. *Scikit Learn k-means*. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html [Accessed: 27-06-2024].

[SI+18] Michael Stonebraker, Ihab F Ilyas et al. 'Data Integration: The Current Status and the Way Forward.' In: *IEEE Data Eng. Bull.* 41.2 (2018), pp. 3–9.

[Zoh11] Erhard Rahm Zohra Bellahsene Angela Bonifati. *Schema Matching and Mapping*. 2011.