GraLMatch: Matching Groups of Entities with Graphs and Language Models

Fernando De Meer Pardo
fernando.demeerpardo@zhaw.ch
University of Zurich
Zurich, Switzerland
Zurich University of Applied Sciences
Winterthur, Switzerland

Claude Lehmann
claude.lehmann@zhaw.ch
Zurich University of Applied Sciences
Winterthur, Switzerland

Dennis Gehrig
dennis.gehrig@zhaw.ch
Zurich University of Applied Sciences
Winterthur, Switzerland

Andrea Nagy
andrea.nagy@movedigital.ch
Move Digital AG
Zurich, Switzerland

Stefano Nicoli
stefano.nicoli@movedigital.ch
Move Digital AG
Zurich, Switzerland

Branka Hadji Misheva
branka.hadjimisheva@bfh.ch
Bern University of Applied Sciences
Bern, Switzerland

Martin Braschler
martin.braschler@zhaw.ch
Zurich University of Applied Sciences
Winterthur, Switzerland

Kurt Stockinger
kurt.stockinger@zhaw.ch
Zurich University of Applied Sciences
Winterthur, Switzerland

ABSTRACT

In this paper, we present an end-to-end multi-source Entity Matching problem, which we call *entity group matching*, where the goal is to assign to the same group, records originating from multiple data sources but representing the same real-world entity. We focus on the effects of *transitively matched records*, i.e. the records connected by paths in the graph $G = (V, E)$ whose nodes and edges represent the records and whether they are a match or not. We present a real-world instance of this problem, where the challenge is to match records of companies and financial securities originating from different data providers. We also introduce *two new multi-source benchmark datasets* that present similar matching challenges as real-world records. A distinctive characteristic of these records is that they are regularly updated following real-world events, but updates are not applied uniformly across data sources. This phenomenon makes the matching of certain groups of records only possible through the use of transitive information.

In our experiments, we illustrate how considering *transitively matched records* is challenging since a limited amount of false positive pairwise match predictions can throw off the group assignment of large quantities of records. Thus, we propose *GraLMatch*, a *method that can partially detect and remove false positive pairwise predictions through graph-based properties*. Finally, we showcase how fine-tuning a Transformer-based model (DistilBERT) on a reduced number of labeled samples yields a better final entity group matching than training on more samples and/or incorporating fine-tuning optimizations, illustrating how precision becomes the deciding factor in the entity group matching of large volumes of records.

1 INTRODUCTION

Due to remarkable technological advancements in recent decades, data has assumed a pivotal role within today's information society. As the volume of data experiences exponential growth [27], data integration, the process of aggregating and cleaning data originating from different sources, has become highly demanding. In fact, the most time-consuming aspect of the entire data science pipeline is often the cleaning and preparation of data [4, 11].

A vital part of data integration is *Entity Matching* (EM). EM describes the problem of determining whether two or more records refer to the same real-life entity. If both records have the same structure i.e., share unique identifiers, this is a trivial task. However, in the instances in which the structure differs significantly, the EM tasks become increasingly complex. Earlier approaches to address this challenge rely on rule-based algorithms and heuristics [13, 20]. More recently, state-of-the-art systems rely on Machine Learning (ML) systems and are Transformer-based [2, 31].

EM is a computationally intensive process, as given the matching of n records there are $\binom{n}{2}$ possible pairwise matches to consider. Most state-of-the-art systems and benchmarks perform matching of 2 data sources and thus focus on evaluating pairs of records as potential matches. In general, however, Entity Matching requires considering not only a set number of record pairs but also all *transitively matched* records.

We consider records r_i and r_j to be *transitively matched* by a given pairwise matching logic if there exists a path between them in the graph $G = (V, E)$ whose nodes and edges represent the records to be matched and the predicted pairwise matches of a pairwise matching logic respectively. All *transitively matched* records are implied to be matches. The expected output of a matching is thus a list of *groups of records* represented as *complete graphs* whose nodes and edges represent records and matches (both predicted and transitive), respectively.

Note that transitively matched records can appear in *any matching setting*, no matter the number of data sources involved. Consider, for example, the matching of data sources S1 and S2 with records A & B belonging to S1 and record C belonging to S2.

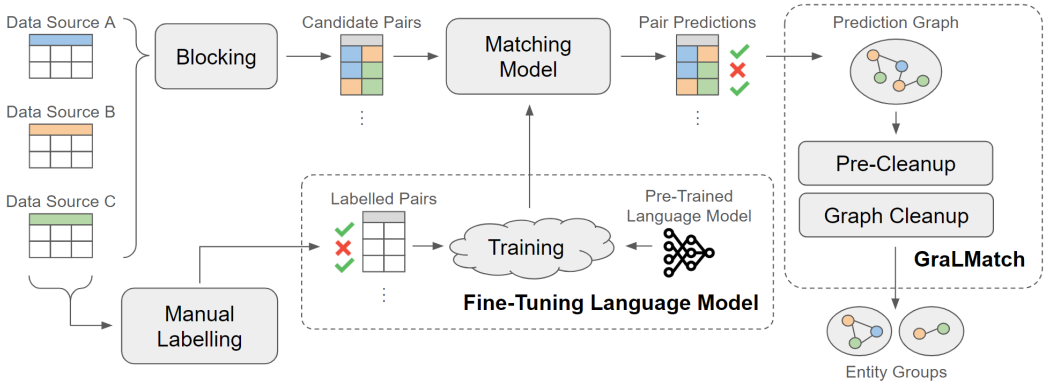

Figure 1: Illustration of the workflow of our entity group matching methodology.

If a pairwise matching logic predicts matches [A-C] and [B-C], then records A & B will be transitively matched, i.e. the transitive match [A-B] is implied by the two previous pairwise matches. Current pairwise matching approaches do not take transitive matches into account and thus ignore the group assignments implied by their pairwise predictions.

In this paper, we refer to the end-to-end multi-source EM problem as *entity group matching*, and propose a novel matching methodology that combines state-of-the-art pairwise EM models and *GraLMatch*, a Graph Cleanup technique that detects and removes false positive pairwise predictions to produce groups of records. Starting from the raw source tables, we apply a blocking to select a subset of candidate record pairs that we predict as either matches or non-matches through some pairwise matching model. At this stage we apply GraLMatch, a Graph Cleanup method that processes all the pairwise predictions, removes those likely to be false positives and produces groups of records. Figure 1 provides a visual representation of the entity group matching methodology we propose, which is not limited to language model-based pairwise matching models, but also supports any matching method that produces pairwise matches.

We focus on a real-world instance of *entity group matching*, where the challenge is to match records of companies and financial securities. The records we want to match are provided by financial data vendors and thus represent licensed information which cannot be readily shared. To overcome this, we generate two *new multi-source synthetic benchmark datasets* that aim to mimic the characteristics of the original financial records. Starting from a publicly available set of records, we carry out a series of algorithmic modifications in order to create groups of companies and securities records that present similar matching difficulties as real-world examples.

A distinctive characteristic of the datasets that we target with our methodology compared to other commonly used benchmarks for EM is that their records are regularly updated reflecting real-world events[1] that may not be recorded equally across data sources. This *data drift* phenomenon, along with instances of missing data, makes some matches between altered records impossible to identify based on their individual attributes alone. As stressed before, this phenomenon calls for methods capable of leveraging transitive information among records.

An illustration of the entity group matching problem is given in Figure 2 where the companies and securities records come from four different data sources. These record groups illustrate some of the different matching challenges present both among real records and our synthetic benchmarks. Some record groups can be matched via the identifiers of their corresponding securities. However, matching identifiers do not guarantee a correct match due to *data drift*. Other groups need to be matched by *text alignment*, i.e. recognizing records as matches via their textual attributes. However, trying to find such groups unavoidably leads to false positive matches and thus to incorrect *transitively matched records*. We deal with these false positives via *GraLMatch*.

Overall, the aim of this paper is to introduce the *entity group matching* task and the concept of *transitively matched* records. We evaluate the performance of state-of-the-art pairwise EM algorithms on challenging datasets, illustrate the effect of transitively matched records and investigate whether our novel algorithm GraLMatch, can be combined with the pairwise algorithms to achieve good entity group matching results. Consequently, the main contributions of our paper are as follows:

(1) We introduce the entity group matching task and illustrate its comparatively higher difficulty w.r.t pairwise matching due to transitively matched records. In our experiments, we illustrate how the evaluations of pairwise EM algorithms carried out in previous works are lacking in terms of entity group matching. To the best of our knowledge, this is the first paper to introduce the problem of matching entity groups, and thus lays the foundations for novel research challenges.
(2) We introduce two new challenging synthetic multi-source benchmark datasets for entity group matching, inspired by a real use case.
(3) We present GraLMatch, a method that addresses the transitivity problem and uses graph-based techniques to detect and remove false positive pairwise predictions that lead to incorrect transitively matched records.

The remainder of the paper is organized as follows. In Section 2 we discuss the related work. Section 3 introduces the synthetic datasets we generate and the real-world use case that inspires them. Section 4 describes the entity group matching problem with a focus on *transitively matched records* and our proposed methodology to deal with them. Section 5 describes the datasets, models and the scores of our experimental setup. Section 6 discusses the results of the experiments and Section 7 outlines our conclusions.

[1]Such as rebrandings, mergers & acquisitions, new stock listings, bankruptcies, etc.

# ID	Name	Location	Description	Data Source
10	lastminute.com group	Chiasso, Ticino, Switzerland	Lastminute is a publicly traded multinational Group, among the world leaders in the online travel industry.	1
11	Hero Telecoms (Pty) Ltd	Stellenbosch, South Africa	Hero Telecoms provides communication services, wireless broadband for homes and businesses.	1
12	CrowdStrike Plt.	Sunnyvale, CA, USA	CrowdStrike Plt. is a cybersecurity technology firm pioneering cloud delivered protection across endpoints, cloud workloads, identity and data.	1
13	Crowdstreet Holdings	Austin, Texas, USA	CrowdStreet is a provider of online commercial real estate investment offerings and technology connecting investors with CRE firms.	1
20	lastminute.com	TI, CHE	-	2
21	Herotel Limited	SA	herotel.com -- acquired by Hearst	2
22	Crowd Strike Platforms	TX, USA		2
23	Crowd street	TX, USA	-	2
30	Last Minute Group	CH	Last Minute Group is an online travel and leisure retailer.	3
31	Crowdstrike Holdings		Cloud-delivered cybersecurity provider.	3
32	CROWDSTREET		Online private equity real estate investing platform.	3
33	Hearst	US		3
40	Crowdstrike Holdings Inc.	US		4
41	HEARST	US	Mass media and business information conglomerate.	4
42	Travix	Amsterdam	Travix is a global online travel company.	4

Name	ISIN	CUSIP	VALOR	SEDOL	Company # ID	Data Source
lastminute.com Ordinary Share	CH440167484U	P5KSVGHI0		SQZW6X	10	1
Hero Telecoms (Pty) Ltd Ordinary Share	ZA6AYIJ89219	K1RQXL44H	107102908		11	1
CrowdStrike Plt. Ordinary Share	US318077556E	318077DSI	109790723		12	1
Crowdstreet Holdings Ordinary Share	US11OQ943600	11OQ94360	117873624		13	1
Equity Shares	CH213423553U				20	2
Equity Shares	US6939793114	K1RQXL44H			21	2
Equity Shares	US318077DSIE				22	2
Equity Shares	US11OQ943600			L9HAA4	23	2
Common Stock	CH440167484U		122290492	L8KQUZ	30	3
Common Stock	US318077556E				31	3
Common Stock	US11OQ943600			L9HAA4	32	3
Common Stock	US6939793114	693979311	124831943		33	3
Common Equity	US318077DSIE				40	4
Common Equity	US6939793114	693979311		N1CO31	41	4
Common Equity	NL6944481TED		122290492	L8KQUZ	42	4

Figure 2: An example dataset of companies (top part) and securities records (bottom part) to match across multiple data sources. Records #12, #22, #31 and #40 correspond to the same entity, "Crowdstrike". Record #12 can be matched to #31 because they have securities with a matching ISIN, US31807756E highlighted in orange. Equivalently with #22 and #40 with US318077DSIE highlighted in violet. Matching the entire group however, requires recognizing all of the different naming variations as equivalent (Crowdstrike Plt./Crowd Strike Platforms/ Crowdstrike Holdings etc.). This task is not trivial, since many false positive predictions are likely to happen with, for example, records #13, #23, #32 corresponding to the entity "Crowdstreet", due to the long shared character sequences across records.

2 RELATED WORK

Early approaches to EM tackled the problem through combinations of hand-crafted features and rule-based heuristics [13, 20] that struggle when faced with dirty/unstructured records. The advent of NLP techniques, more specifically Transformer models, has led to the current state-of-the-art EM systems which frame EM as a ML problem and treat records as instances of textual data.

Transformer Language Models are a class of neural network models based on the Transformer architecture [31] that have been shown to excel at numerous NLP tasks [9]. These models are pre-trained on large text corpora in order to acquire natural language understanding via self-supervised tasks. In order to perform downstream tasks, one can perform Transfer Learning by adding task-specific final layers to the Transformer model initialized with the pre-trained weights and then fine-tune it on a task-specific training set until convergence, usually for a small number of epochs.

BERT (Bidirectional Encoder Representations from Transformers) and its variants [5, 16, 28] are a class of models consisting of the Encoder part of the Transformer architecture. They have been shown to achieve state-of-the-art pairwise matching results at numerous EM benchmark datasets by following the fine-tuning process mentioned above along with different optimizations specific to the EM task such as Data Augmentation and Contrastive Learning [2, 15, 17, 22, 30, 33, 34] that aim to tackle the specific challenges presented by EM: (1) Lack of labeled training pairs, and (2) the difficulty of obtaining quality embeddings for nuanced similarity-based discriminative tasks, given the ambiguity present between many records. Additionally, optimizations explored for *multi-source Entity Matching* include combining binary and multi-class classifications [21] and active learning using the graph implied by pairwise matches for query selection and data augmentation [25].

All previous works, however, only evaluate models according to their performance at pairwise matching, ignoring transitive matches. Compared to the setting with only 2 data sources, transitive matches can be expected to have a greater effect in matching settings with multiple data sources[2]. However, as we discussed in the previous section, transitively matched records can appear in *any matching setting*. Previous models have only been evaluated according to their performance on pairwise matching in set data splits, which is a much easier task than entity group matching.

[2]Simply because greater numbers of candidate matches are evaluated and thus false positive pairwise predictions are more likely.

In our experiments, we will show how a close-to-perfect pairwise matching performance is not sufficient to achieve a good matching if transitive matches are not considered.

More recently, Transformer architectures have been scaled to billions of parameters, leading to a new class of models known as *Large Language Models* (LLMs). These models are trained in the same self-supervised way as their smaller counterparts but are additionally further tuned via *Reinforcement Learning with Human Feedback* (RLHF), which incorporates human input into the training process in order to align the model's behaviour with human preferences. LLMs have been shown to be capable of carrying out pairwise matching by framing it as a question-answering problem [19, 23]. While promising, these models are considerably slower at inferencing and thus unsuitable for matching large datasets such as the ones we propose, as this requires millions of pairwise match evaluations to be carried out.

Many benchmark datasets have been used over the years to compare the performance of newly proposed EM methodologies against existing ones. Older benchmarks mostly contain structured records of 2 data sources [6, 14, 18], while more recent ones often present records consisting only of textual descriptions originating from more data sources, which go into the thousands for web-scraped records [24]. Additionally, multiple Data Generators that produce datasets of varying difficulty by polluting certain records (e.g. via modification or deletion of certain attribute values) have been also proposed [3, 8, 10, 12, 26, 29, 32]. We add to this line of research by proposing a multi-source matching problem that requires leveraging transitive information.

3 ENTITY GROUP MATCHING PROBLEM: A REAL-WORLD USE CASE

In this section, we introduce a use case of entity group matching based on the integration of financial records originating from various real-world data sources. This section serves as the basis for the generation of *two new entity group matching benchmark datasets*. First, we describe the origin and nature of the *original data*, which, due to privacy regulations, cannot be shared with the research community. Next, we describe the generation of the *synthetic dataset*, which contains the most important characteristics of the original data and can be openly shared.

3.1 Original Data

The original real-world datasets contain records consisting of a series of textual and alphanumerical attributes for *companies*[3] and *securities*[4] that are traded on certain stock exchanges. Companies can issue multiple securities, but each security belongs exclusively to a single company.

As previously mentioned, these records are continuously updated reflecting real-world events such as bankruptcies, mergers, acquisitions, rebrandings etc. However, updates are not carried out uniformly across all data sources. Consequently, the records across our data sources not only differ due to variations in naming practices (such as using the full company name "Microsoft Corporation" or the stock ticker "MSFT") but also because of the events described above, which are typically absent in other EM benchmarks. In our use case, we have records from around 10 different data sources, i.e. financial data providers, such as Bloomberg, Reuters, etc.

In order to ease the matching of both company and security records, several international identifier standards have been developed such as International Securities Identification Numbers (ISINs) or Legal Entity Identifiers (LEIs). However, groups of records affected by real-world events are not recognized and/or catalogued by matching approaches relying merely on these identifiers. For example, if a match is made between two companies based exclusively on their LEIs, we might be incorrectly matching an acquirer with its acquiree after an acquisition that led to the overwriting of the LEI of the latter with that of the former. Additionally, many records are missing some or all identifiers and thus can only be matched based on their textual attributes, such as record #20 in Figure 2, which calls for sophisticated EM methods.

3.2 Synthetic Benchmark Datasets

The original datasets are offered by data vendors and distributed under non-disclosure agreements which prevent their open distribution. To overcome this privacy issue, we generate a *synthetic benchmark dataset* from a publicly available set of 1.04M company records provided by Crunchbase[5], which serves as a starting point for recreating our real-world dataset. From the Crunchbase dataset, we extract the *name, city, region, country_code, and short_description* attributes of the first 200K records. In order to create company records that present the same kind of variability as observed among our real-world records, we implement a series of algorithmic modifications of the extracted records that we name *data artifacts*. These *data artifacts* modify the original Crunchbase records via rule-based algorithms, much like the data augmentation operators used in pseudo-labeling methods [15, 33]. Examples of data artifacts include:

(1) **AcronymName (Companies):** Swap a name with its acronym.
(2) **InsertCorporateTerm (Companies):** Inserts a common term (Inc./Limited/Corp etc.) in all mentions of a name.
(3) **CreateCorporateAcquisition/Merger (Companies):** Overwrites records with the attributes of an acquiring investor simulating an acquisition/records the interaction with another company simulating a merger process.
(4) **ParaphraseAttribute (Companies):** Paraphrase a textual attribute via the Pegasus summarization model.
(5) **MultipleIDs (Securities):** Create new identifiers and assign them to multiple records of a security.
(6) **NoIdOverlaps (Securities):** Wipe all overlaps among identifiers of a group of security records.
(7) **MultipleSecurities (Securities):** Adds new securities of different types such as rights, bonds or units to an issuing company.

Each of the record groups of the synthetic datasets will have a random combination of *data artifacts* applied to it. Note that multiple *data artifacts* are sequentially applied to each record group and thus their effects become intertwined, generating a big variety of matching challenges across the 200K groups.

[3]Typical attributes for companies are: company names, textual descriptions, headquarter addresses, industry classification codes, market capitalization, # of employees, etc.

[4]Typical attributes for securities are: security names, types and unique identifiers based on (inter)national standards such as ISINs (**I**nternational **S**ecurities **I**dentification **N**umber), CUSIPs (**C**ommittee on **U**niform **S**ecurities **I**dentification **P**rocedures), VALORs (Swiss German banking term for a "security"), SEDOLs (**S**tock **E**xchange **D**aily **O**fficial **L**ist), etc.

[5]See Crunchbase's Basic Export: https://data.crunchbase.com/docs/crunchbase-basic-export

The size of the generated dataset[6] and the proportion of record groups to which each *data artifact* is applied to, can be fully parameterized and different choices of generation parameters will lead to datasets with different sizes and proportions of record groups presenting each type of matching challenge. We have calibrated the generation by inspecting the main characteristics of the real dataset[7]. Our specific choice of parameters, along with instructions on how to modify them to carry out different generations, is described in the accompanying code[8].

In a real setting, the extent to which different records affected by real-world events should be matched, will vary on a case-by-case basis. Mergers usually involve the creation of a new entity with shares of the original companies being exchanged based on an agreed-upon exchange ratio, which will be different for each of the merging companies depending on their finances. Often, no records are deleted due to a merger, but rather, records of the new merged entity are created. We do *not* consider records involved in simulated mergers as matches.

On the other hand, acquisitions involve one company absorbing the other, with shareholders of the target company receiving compensation for their shares, which cease to exist along with the absorbed company, whose records are usually deleted in the data sources that record the event. Since the acquiring company has to assume the finances of the acquiree, we consider all records involved in acquisitions as matches.

Records involved in mergers and acquisitions may have some or all of their attributes overwritten due to the event. In Figure 2 Records #10, #20, #30 corresponding to "lastminute.com" and #42 corresponding to "Travix" are involved in a merger and thus #30 has had some identifiers overwritten with those of #42 while not being a true match. Records #11 and #21 corresponding to "Herotel" and #33, #41 corresponding to "Hearst" are involved in an acquisition and thus are all matches. Note that the identifiers of the security record #21 have been overwritten with those of #33 and #44.

3.3 Real vs. Synthetic Dataset

While far from being able to generate all the variations and interactions present among real records[9], randomly combining *data artifacts* leads to a *novel, non-trivial dataset for entity group matching*. Below, we describe the main challenges of the dataset:

(1) Record pairs with **matching identifiers** are not necessarily true matches since an alignment between textual attributes is needed to detect records affected by **mergers** or **acquisitions** events, which may have had some or all of their attributes overwritten.
(2) Matches between records **lacking matching identifiers** can only be found by matching their textual attributes such as record #20 of Figure 2 which has a different identifier than the one records #10 and #30 share and thus has to be matched via its name.
(3) Matches between records whose **identifiers and textual attributes** are different (e.g. the records of an acquiring company and the acquiree when the data source of the latter has not recorded the acquisition), may only be found by taking into account *transitive information* found through other matched records or be otherwise impossible to detect. For instance, records #11, #33 and #41 of Figure 2 can only be matched by matching record #21 to #33 or #41 first.

[6] A maximum of 1.04M (size of the Crunchbase dataset) record groups can be generated. Data artifacts are applied sequentially to each record group thus the complexity of the generation is linear w.r.t the number of record groups.
[7] We choose generation parameters based on observations made on the subset of manually labeled real records that we use in the experiments. Note that fully characterizing the proportion of matching challenges present in the record groups of the real dataset would require both being able to perfectly match the dataset (i.e. the objective of this work) and then manually annotating the origin of all the challenges each group presents. The latter, however, is an ambiguous task since combinations of record updates due to real-world events, much like *data artifacts*, often lead to matching challenges whose causes are impossible to discern.
[8] See the README file in the folder `datainc/datainc_code`: https://github.com/FernandoDeMeer/GraLMatch
[9] Examples of additional difficult phenomena to generate algorithmically while guaranteeing matchability are multiple languages/alphabets, rebrandings, geographical terms, etc.

Table 1: Overview of the general statistics of our datasets. Values marked with a *-symbol are estimated, as only a subset of the real datasets were manually checked and labelled.

Dataset	Companies		Securities	
	Real	Synthetic	Real	Synthetic
# of Data Sources	~ 10	5	~ 10	5
# of Entities	$< 200K^*$	200K	$< 250K^*$	$\sim 275K$
# of Records	$\sim 600K$	868K	$\sim 1M$	$\sim 984K$
# of Matches	$> 1M^*$	$1.5M$	$> 1.5M^*$	$\sim 1.5M$
Avg. # of Matches per Entity	7^*	7.5	10^*	~ 5.4
% of Records with Text Descriptions	25%	32%	-	-

See Table 1 for a comparison between the generated (synthetic) datasets and their real counterparts.

4 ENTITY GROUP MATCHING WITH GRALMATCH

In this section, we introduce our novel *entity group matching* approach that enables the matching of record groups – as opposed to only matching record pairs, which is the common approach of previous Entity Matching research. Note that simply using pairwise matching to solve the entity group matching problem is not sufficient, as we will highlight in this section.

As stated in Section 1, Entity Matching requires considering the *transitively matched* records of a given pairwise matching logic, even if the pairwise matching model did not directly predict each pair of *transitively matched records* to be a match. Including the implied transitive matches of a given set of pairwise match predictions is necessary to obtain each of the expected groups of records (complete graph) of the entities we intend to match.

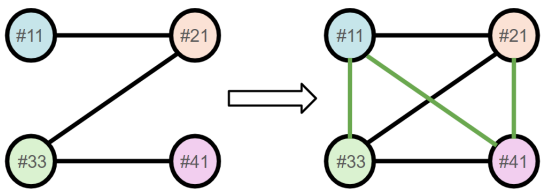

Figure 3: Example of transitive matches between records of Figure 2. On the left side, the pairwise matches (#11 and #21), (#21 and #33) and (#33 and#41) imply the transitive matches of the right side colored in green (#11 and #33), (#11 and #41) and (#21 and #41).

Figure 3 shows an example of a group of records, identified by the #ID attribute as in Figure 2, with pairwise matches that

imply transitive matches. Note that only record #21 contains information about the relationship between all records of the group (the acquisition) and thus the group can only be discovered by matching #21 against either record #33 or #41 and finding all other matches transitively.

Conversely, Figure 4 shows an example of 2 different record groups incorrectly linked through a false pairwise match prediction between records #40 and #13. These predictions greatly affect the matching because they lead to all records of both groups being *transitively matched*, which results in numerous false transitive matches.

Our end-to-end matching methodology is illustrated in Figure 1 and addresses the transitivity problem through the following steps:

(1) **Fine-tune Language Models**: We fine-tune a Language Model to perform Sequence Classification in order to predict record pairs as either `Match` or `NoMatch`. We focus on Language Models because they are the state-of-the-art pairwise matching methods.
(2) **Pairwise matching:** We evaluate a set of record pairs obtained via some blocking[10] with the fine-tuned Language Model.
(3) **GraLMatch Graph Cleanup:** Considering the graph $G = (V, E)$ whose nodes and edges represent the records to be matched and the positively predicted matches of the previous step, we delete a series of edges/matches of G with a method that considers the graph implied context (i.e. the connected component it lies on) of each edge/match.
(4) **Entity Groups:** We output all the connected components of the updated graph as groups of matched records.

4.1 Pairwise matching with LMs

Transformer-based Language Models offer several desirable properties for performing pairwise matching over traditionally used heuristics. Namely, through techniques such as *tokenization, word embeddings, attention mechanisms and pre-training* they are able to jointly process attributes of different data types, either structured or unstructured, making context-aware predictions learned from fine-tuning on large volumes of data.

We employ DistilBERT [28] for our experiments, a model obtained via distillation of the pre-trained BERT model. While having 40% fewer parameters than BERT, DistilBERT retains close language understanding capabilities w.r.t. BERT, as evidenced by the similar scores achieved at a series of downstream tasks. Due to the scale of the datasets we aim to match, we choose DistilBERT for faster evaluation speeds given the vast number of comparisons required for the pairwise matching step.

We fine-tune the pre-trained DistilBERT on the binary classification task of predicting record pairs as either `Match` or `NoMatch` with a subset of matches and non-matches from the training dataset. In a real setting, only a subset of labeled matches may be available since the labeling effort must be considered. Once the training pairs are decided, we add a final softmax layer to the pre-trained model and then train it for a few epochs as in [2, 15].

4.2 GraLMatch Graph Cleanup

Consider the graph $G = (V, E)$ whose nodes and edges are given by the records to be matched and the predicted matching pairs, respectively. The aim of the GraLMatch Graph Cleanup step is to recognize false positive pairwise match predictions to remove them from the final output. The false positives are removed using the *graph implied context* of each pairwise prediction, i.e. the connected component it lies on, or equivalently, all the records *transitively matched* to both records of the pair. Different approaches can be employed to discover good candidate edges for removal, depending on the type of information considered from the connected component.

False positive predictions are often the only link between sets of densely connected nodes such as the match between records #40 and #31 of Figure 4. The following graph-based methods allow us to filter such edges and mark them for removal:

(1) *Minimum Edge Cut:* Given a graph $G = (V, E)$ a *minimum edge cut* S is a set of edges $S \subseteq E$ of minimum cardinality (i.e. $|V_S|$ if $S = (V_S, E_S)$) that disconnects G.
(2) *Edge Betweenness Centrality:* Given a graph $G = (V, E)$, the Betweenness Centrality of an edge $e \in E$ is the sum of the fraction of all-pairs shortest paths that pass through e:

$$c_B(e) = \sum_{s,t \in V} \frac{\sigma(s, t \mid e)}{\sigma(s, t)}$$

where $\sigma(s, t)$ is the number of shortest-paths between nodes s and t, and $\sigma(s, t \mid e)$ is the number of those paths passing through edge e.

The matching of large datasets may lead to numerous large connected components, each potentially containing multiple good candidate edges for removal. Note that removing the edges of a connected component belonging to a *Minimum Edge Cut* guarantees disconnecting said component, while the same is not true when removing the edges with highest *Edge Betweennes centrality*. Both techniques have a time complexity of $O(mn)$ where n is the number of records and m the number of edges of a given connected component [1, 7]. However, in practice, the *Minimum Edge Cut* tends to have a lower run-time, even if both worst-case time complexities are identical. We generally expect the *Edge Betweennes centrality* to remove less true positive edges while being slower than the *Minimum Edge Cut*.

Algorithm 1 shows how the two techniques described above are used by our GraLMatch Graph Cleanup. Size threshold γ specifies which of the techniques should be used, depending on the size of the connected component to clean up, whereas μ specifies the desired maximum size of all produced record groups.

We set size threshold μ to be equal to the number of data sources. This is ideal in settings where each group is expected to have at most one record per data source, as is the case in our datasets of interest. However, in settings where there is not a specified number of data sources and/or we expect a lot of record groups of different sizes, as in the case of web-scrapped records, Algorithm 1 will not be ideal and other Graph Cleanup methods able to produce groups of heterogeneous sizes should be considered.

4.2.1 Pre Graph Cleanup. Some sets of pairwise match predictions lead to exceedingly large connected components in G. In order to avoid long running times of Algorithm 1[11], we further apply the following pre-cleanup technique:

Company datasets: We remove all positively predicted matches obtained through the *Token Overlap* blocking in connected components larger than 50 records. See Section 5.3.1 for details.

[10] Blocking is necessary due to the large number of possible candidate pairs and the impossibility of evaluating all due to prohibitive running time.

[11] Both of the edge removal techniques used during the GraLMatch Graph Cleanup remove a few edges at a time and are thus slow at cleaning up exceedingly large connected components.

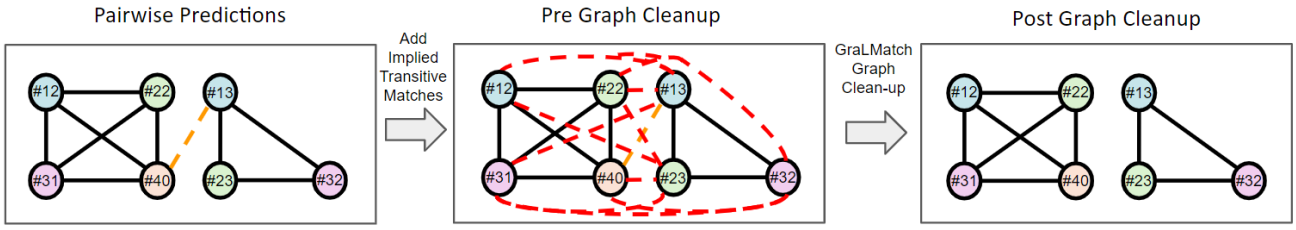

Figure 4: Illustration of entity group matching based on a subset of the records shown in Figure 2. (1) Pairwise predictions: The false positive pairwise match between record #40 (Crowdstrike) and record #13 (Crowdstreet) is illustrated as a dotted orange line. (2) Pre Graph Cleanup: False transitive matches are shown as dotted red lines, e.g. record #12 (CrowdStrike) is wrongly matched transitively with record #13 (CrowdStreet). True positive pairwise and final matches are black lines. (3) Post Graph Cleanup: The false pairwise match, originally shown in orange, is eliminated via the GraLMatch Graph Cleanup. The results are two group matches as opposed to one group match resulting from wrong pairwise matching.

Algorithm 1 GraLMatch Graph Cleanup

Input: matches graph $G = (V, E)$, size thresholds γ and μ
Output: List of connected components of G after cleanup.

$C = \{c_1, c_2, ..., c_n\}$ connected components of G
$c^* \leftarrow \arg\max(\{\,|c_i| \mid c_i \in C\})$
while $|c^*| > \gamma$ **do**:
  $E_{\text{mincut}} \leftarrow \text{MinEdgeCut}(c^*)$
  $G \leftarrow (V, E \setminus E_{\text{mincut}})$
  $c^* \leftarrow \arg\max(\{\,|c_i| \mid c_i \in C\})$
while $|c^*| > \mu$ **do**:
  $e_{\text{maxBC}} \leftarrow \arg\max(\{BtwCent(e) \mid e \in E^*\})$
  $G \leftarrow (V, E \setminus e_{\text{maxBC}})$
  $c^* \leftarrow \arg\max(\{\,|c_i| \mid c_i \in C\})$
Output: $C = \{c_1, c_2, \ldots, c_n\}$

5 EXPERIMENTS

In this section, we describe our experimental setup[12]. The goal is to address the following research questions:

- What is the *performance* of state-of-the-art Entity Matching algorithms for *pairwise and entity group matching* applied to our challenging datasets?
- Can our *novel algorithm GraLMatch boost the performance existing algorithms for entity group matching*?

All the experiments run in this paper were conducted on an Ubuntu machine with a Nvidia Tesla T4 GPU, 16 VCPUs, and 64 GB of RAM.

5.1 Datasets

We now describe both the real and synthetic datasets used for our experiments. Our aim is to *match multiple real-world datasets* made up of records of companies and financial securities. As described in Section 3, due to the confidential nature of this data, we have produced a dataset of groups of records from publicly available data that present similar matching challenges as their real counterparts.

5.1.1 Real Companies and Securities Datasets. In the experiments, we use a small subset of 63.5k human-labeled security record groups obtained through matching identifier codes combined with 1.5k manually found edge case record groups (with missing identifiers, multiple identifiers for matching securities, mergers, acquisitions etc.), totaling to 65K companies and securities from 8 different data sources. For the company records, we use the corresponding issuers of each of the securities. These datasets are small in relation to the entire real dataset and contain a very low proportion of challenging record groups, due to the difficulty involved in manually finding them, see Table 1 for reference.

5.1.2 Synthetic Companies and Securities Datasets. We perform experiments on the synthetically generated datasets described in Section 3 in order to estimate the different models' performance at entity group matching in a real world setting. These datasets are closer in scale to the entire real datasets than the human-labeled set of real records described above, see Table 1 for reference. Consequently, the results we obtain in these datasets will give us a better approximation to our performance in the real use case which involves the entire real dataset. We make these datasets, as well as the code to generate them, fully accessible to the wider research community for utilization.

5.1.3 Train, Validation and Test Splits. In order to fine-tune and evaluate each machine learning model, we divide the records of the datasets above into *train, validation and test* splits, each containing all the records belonging to 60%/20%/20% of the ground truth record groups[13]. We fine-tuned models with all the positive pairs of each split and add randomly sampled negative pairs with a ratio of 5 : 1 negative pairs for each positive one. We split along the record groups to make sure that the set of true matches of each entity belongs exclusively to one split, preventing models from memorizing pairs.

5.1.4 WDC Products. Additionally, we evaluate our matching pipeline on the WDC Products benchmark dataset [24]. More specifically, we experiment on the large dataset variant with 80% corner cases and a test set with 100% unseen entities.

5.2 Machine Learning Models

For our experiments we use Ditto [15], the state-of-the-art machine learning model for Entity Matching, as our baseline. We also considered employing a large language model, LlaMa2 7B, by framing the classification task as a text generation instance via prompt-engineering and recovering answers via a regular expression. Initial experiments showed that LlaMa2 took on average 7 seconds to generate an answer per candidate pair which leads to

[12]Our code can be found here: https://github.com/FernandoDeMeer/GraLMatch

[13]The percentages roughly correspond to the % of records in each split but there are small variations because record groups vary in size.

exceedingly long running times for the pairwise matching step (90+ days). We expect this also to be the case with comparably sized and bigger large language models. We use the following specifications for each model:

- **DITTO (128)**: In accordance with our focus on speed, this DITTO [15] variant uses the DistilBERT [28] model due to the size of our datasets, similar to the choice in the original DITTO paper for their larger datasets. Additionally, the DITTO (128) variant uses a max sequence length of 128 tokens, half the size used in the original paper. Please note, that DITTO uses a different encoding scheme which wraps column names and values with special start and end tokens. For example, the value `Zurich` in the `city` column would be encoded as `[col] city [val] Zurich`. This increases the amount of tokens required to encode the same value information, but adds more structure and the information of column names.
- **DITTO (256)**: This variant of DITTO is identical to the DITTO (128) variant, except that it uses sequences with up to 256 tokens (as in the original paper).
- **DistilBERT (128)-ALL**: We fine-tune DistilBERT with a maximum token sequence length of 128 on all of the pairs in the train splits.

5.2.1 Sensitivity Analysis. In order to study the sensitivity of our pipeline w.r.t the amount of available labeled data, we fine-tune DistilBERT (128) on small sets of training pairs from the synthetic companies and securities datasets that could be acquired with a moderate manual labeling effort:

- **DistilBERT (128)-15K**: We fine-tune DistilBERT with a maximum token sequence length of 128 on a set of pairs obtained by *filtering the first 10K/5K pairs from the train/val splits*. We discard those whose records have been involved in an *acquisition* or cannot all be matched via identifier overlaps.

Additionally, in order to study the sensitivity w.r.t the size thresholds γ and μ, we run the matching pipeline with **DistilBERT (128)-ALL** and the following modifications:

- **DistilBERT (128)-ALL-MEC**: We run the Graph Cleanup with $\gamma = \mu$, that is, we only use the *Minimum Edge Cut (MEC)* to remove edges.
- **DistilBERT (128)-ALL ($\frac{1}{2}\gamma$)**: We set the size threshold γ to half of its value in Table 2 (rounded down).
- **DistilBERT (128)-ALL-BC**: We run the Graph Cleanup with $\gamma = \infty$, that is, we only use the *Betweenness Centrality (BC)* to remove edges.

We run the 3 previous variants on the Synthetic Companies dataset, since it is the dataset with the biggest number of candidate pairs (and thus connected components), see Table 2. For all models, we fine-tune for 5 epochs and select the epoch with the lowest validation loss.

5.3 End-to-End entity group matching Experiments

As mentioned previously, our aim is to *match multiple real-world datasets* for which a ground-truth is initially unavailable. To estimate our performance on this challenging task, we carry out the end-to-end entity group matching process on the datasets described in Section 5.1.

5.3.1 Blockings. In order to start the entity group matching process, we first need to obtain a subset of candidate record pairs through a combination of blockings in order the reduce the complexity and running time of the matching process. We employ the following blockings:

(1) ***ID Overlap:*** Finds candidate pairs based exclusively on the overlap of identifier attributes. Typical examples of IDs for *securities* are ISINs, CUSIPs, VALORs or SEDOLs as shown in the bottom part of Figure 2. In the case of *company* records, we evaluate against the companies whose associated securities have a matching identifier with any of the securities issued by each company record. This blocking is equivalent to the benchmark heuristic often used to match these types of financial records. It leads to few candidate pairs containing positive and negative pairs as described in Section 3.3.
(2) ***Token Overlap:*** Considers each record as the list of tokens resulting from its tokenization and selects as candidate pairs those involving the record and the top n records with most overlapping tokens across different data sources. This blocking aims to find good candidate pairs for matches based on text alignment.
(3) ***Issuer Match (Securities Only):*** For each security record, consider as candidate pairs those involving all other securities issued by companies previously matched to the security's issuer. This blocking allows to match pairs of securities with non-matching identifiers and generic names based on a previous matching of their issuers.

We combine the blockings described above differently for each dataset as detailed in Table 2, producing different sets of candidate pairs. We predict each of the candidate pairs as either `Match` or `NoMatch`, leading to a set of pairwise match predictions for each model/dataset pair.

Table 2: Blockings applied, number of records, candidate pairs and size thresholds in the entity group matching experiment for each dataset.

	Blockings	# of Records	# of Candidate Pairs	γ	μ
Real Companies	*ID Overlap* *Token Overlap*	6.3K	51K	40	8
Synthetic Companies	*ID Overlap* *Token Overlap*	174K	1.14M	25	5
Real Securities	*ID Overlap* *Issuer Match*	12.8K	41K	40	8
Synthetic Securities	*ID Overlap* *Issuer Match*	197K	826K	25	5
WDC Products	Token Overlap	1K	9.1K	25	5

5.3.2 Impact of GraLMatch Graph Cleanup. In order to illustrate the importance of the GraLMatch Graph Cleanup step with respect to the final entity group matching, we calculate *precision, recall and F1* at three different stages of our end-to-end matching pipeline:

(1) ***Stage 1: Pairwise matching:*** The *positively predicted pairs* from all of the candidate pairs produced by the combination of blockings in each dataset.
(2) ***Stage 2: Pre Graph Cleanup:*** Considering the graph $G = (V, E)$ whose nodes and edges represent the records to be matched and the pairwise match predictions respectively, we incorporate all of the edges missing from each connected component to make the component a *complete* subgraph. That is, we add all the edges connecting pairs of *transitively matched* records by the pairwise match predictions.
(3) ***Stage 3: Post Graph Cleanup:*** We run the GraLMatch Graph Cleanup on the graph $G = (V, E)$ (with the pairwise match predictions only) and then add all of the transitive matches to the list of cleaned up connected components produced by the GraLMatch Graph Cleanup.

Note that the scores achieved for *pairwise matching* are not equivalent to those obtained during fine-tuning. The recall is expected to be lower since some *true positives are discarded by the blocking*, whereas all the true positives are available and evaluated during fine-tuning. Additionally, the precision is also expected to be lower since the set of candidate pairs contains more challenging negative pairs than the ones added randomly during fine-tuning.

Considering all of the matches included in the *Pre Graph Cleanup* set is justified because they represent the group assignment implied by the *pairwise matching*. We will show how usually the Pre Graph Cleanup scores are considerably low, since very few false positive predictions can lead to a large amount of false transitive matches, especially when large connected components are produced by a set of pairwise match predictions.

The *Post Graph Cleanup* scores represent the final group assignment of our methodology. See Figure 4 for an illustration of all three sets of matches and the phenomenon described above.

It is important to note that the scores achieved with pairwise matching should not be compared to Pre and Post Graph Cleanup scores since they do not represent a group assignment (they ignore the *transitively matched* records) but rather only represent a model's pairwise matching performance.

5.3.3 Graph Metric. Along with precision, recall and F1 scores, we calculate the following graph-based metric:

- **Cluster Purity Score:** Given an entity group matching M composed of complete subgraphs $\{c_i = (V_i, E_i)\}_{i=1}^{N}$ its cluster purity is calculated as:

$$\text{ClPur} = \frac{1}{\sum_{i=1}^{N} |V_i|} \sum_{i=1}^{N} |V_i| \frac{c_{TP,i}}{|E_i|}$$

  where $c_{TP,i}$ is the number of true positive matches in subgraph c_i.

The Cluster Purity Score is thus the average of the number of correct matches per record group weighted by the size of each group. It indicates how reliable downstream tasks based on the aggregation of calculations made on record groups will be.

6 RESULTS

6.1 Fine-Tuning

Table 3 shows the precision, recall and F1 scores of each fine-tuned model on test split pairs.

On the companies datasets, both real and synthetic, all models reach close-to-perfect scores with the exception of DITTO (128) on real companies, which we speculate misses important elements of some inputs due to its encoding method. DistilBERT(128)-ALL reaches a very similar performance to that of DITTO (256) while using half of the input tokens. DistilBERT(128)-15K achieves a lower recall than all other setups, which is to be expected since it is trained with much fewer samples, yet reaches a competitive F1 score while taking 1/6th of the time to train than the other setups.

On the securities datasets, we similarly observe that DITTO (128) struggles to perform due to its encoding method, this time likely due to the identifier attributes which lead to long sequences of uninformative tokens. We observe that on real securities DistilBERT(128) and DITTO (256) reach a similar F1 score while on the synthetic securities, DITTO (256) reaches the overall best F1 score but struggles with precision. This is likely due to the fact that matches between records whose issuers have been involved in a *data drift* event do not present matching identifiers and have generic names (see for example the securities records #31 and #40 of Figure 2, both belonging to records of the entity "Crowdstrike") but DITTO (256)'s extensive encoding method seems to allow it to capture some of these challenging pairs. DITTO (256) seems more likely to make positive predictions, which captures some *data drift* pairs but also false positives.

On the WDC Products dataset DistilBERT(128)-ALL achieves a slightly worse performance to that of RoBERTa as reported in [24]. The result is expected due to the training optimizations of RoBERTa and its larger model size compared to DistilBERT(128).

DistilBERT(128)-ALL also matches the performance of DITTO (256) while using half of the input tokens and outperforms DITTO (128), which indicates DITTO's encoding and training optimizations do not translate into a better performance in this dataset for the smaller model setup.

6.2 Entity Group Matching with Blocking and GraLMatch

6.2.1 ***Companies Datasets.*** Table 4 shows the precision, recall and F1 scores of the entity group matching experiment of *both real and synthetic companies datasets.*

Let us first analyze the *pairwise matching performances* on pairs produced by blocking (first column of Table 4). For all models, the recall and precision of the pairwise match predictions are slightly lower than those obtained during fine-tuning. The lower recall results from a portion of true pairs not being selected by blocking as candidate pairs. The lower precision is due to the fact that the negative pairs among the candidate matches are much more difficult to classify than the randomly sampled negative pairs used during fine-tuning.

Next, we analyze the *entity group matching performance* (second column of Table 4). As illustrated in Figure 4, false positive pairwise predictions greatly affect the entity group matching performance, since they connect different groups of entities leading to numerous false positive predictions. This phenomenon is reflected by the *Pre Graph Cleanup* Precision and Cluster Purity scores, which evidence the necessity of removing false positive pairwise predictions to achieve a good entity group matching. Additionally, it is more pronounced on the synthetic companies because they are much more numerous (see Table 2) and thus contain considerably more records sharing common terms ("hi-tech", "networks", "energy", "resources", geographical terms etc.)

Table 3: Overview of the scores achieved by fine-tuned models on test pairs.

Dataset	Model	Pairwise Matching Performance Precision	Recall	F1 Score	Training Time
Real Companies	DITTO (128)	68.82 ± 0.00	83.49 ± 0.00	75.11 ± 0.00	**18.74 h**
	DITTO (256)	99.90 ± 0.09	99.67 ± 0.13	**99.78 ± 0.11**	33.59 h
	DistilBERT (128)-ALL	99.93 ± 0.02	99.56 ± 0.04	99.73 ± 0.02	23.25 h
Synthetic Companies	DITTO (128)	99.45 ± 0.06	96.70 ± 0.30	98.15 ± 0.01	85.11 h
	DITTO (256)	99.55 ± 0.01	96.88 ± 0.01	**98.20 ± 0.01**	86.39 h
	DistilBERT (128)-15K	99.35 ± 0.03	94.77 ± 2.74	96.99 ± 1.41	**11.32 h**
	DistilBERT (128)-ALL	99.28 ± 0.13	96.09 ± 0.06	97.66 ± 0.03	93.28 h
Real Securities	DITTO (128)	25.55 ± 7.40	69.00 ± 43.84	33.89 ± 0.17	22.71 h
	DITTO (256)	99.94 ± 0.01	99.13 ± 0.02	**99.53 ± 0.01**	37.88 h
	DistilBERT (128)-ALL	99.48 ± 0.18	99.48 ± 0.14	99.47 ± 0.02	**20.96 h**
Synthetic Securities	DITTO (128)	57.82 ± 2.12	56.00 ± 11.64	56.47 ± 5.01	94.43 h
	DITTO (256)	85.51 ± 0.55	91.35 ± 0.51	**88.33 ± 0.06**	122.44 h
	DistilBERT (128)-15K	94.03 ± 0.41	61.11 ± 0.33	73.26 ± 0.24	**11.62h**
	DistilBERT (128)-ALL	90.96 ± 0.77	70.55 ± 0.52	79.46 ± 0.06	103.99 h
WDC Products	DITTO (128)	35.92 ± 0.01	63.20 ± 2.83	45.81 ± 1.56	27.63 min
	DITTO (256)	48.45 ± 5.39	72.30 ± 7.21	**57.71 ± 1.53**	40.28 min
	DistilBERT (128)-ALL	46.24 ± 0.96	76.33 ± 1.70	57.58 ± 0.96	**26.79 min**

Table 4: Overview of the scores achieved in the *entity group matching with Blocking and GraLMatch. To achieve the Post Graph Cleanup scores* our novel algorithm GraLMatch is applied.

		Pairwise Matching Performance (pairs from blocking)			Entity Group Matching Performance (including implied transitive matches)								
					Pre Graph Cleanup				Post Graph Cleanup				
Dataset	Model	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Cluster Purity	Precision	Recall	F1 Score	Cluster Purity	Inference Time
Real Companies	DITTO (128)	23.66 ± 0.00	99.64 ± 0.00	38.24 ± 0.00	0.05 ± 0.00	99.66 ± 0.02	0.10 ± 0.00	0.00 ± 0.00	99.86 ± 0.12	98.23 ± 0.55	**99.06 ± 0.34**	1.00 ± 0.00	6.7 min
	DITTO (256)	23.66 ± 0.00	99.64 ± 0.00	38.24 ± 0.00	23.52 ± 0.01	99.68 ± 0.04	38.06 ± 0.02	0.00 ± 0.00	98.42 ± 0.01	99.70 ±0.02	99.05 ± 0.01	0.99 ± 0.00	6.6 min
	DistilBERT (128)-ALL	94.06 ± 4.83	99.27 ± 0.24	96.53 ± 2.49	49.07 ± 34.11	99.73 ± 0.16	56.92 ± 38.6	0.80 ± 0.16	86.90 ± 5.07	96.98 ± 3.69	91.64 ± 4.25	0.93 ± 2.50	**3.5min**
Synthetic Companies	DITTO (128)	33.16 ± 0.00	81.73 ± 0.00	47.18 ± 0.00	0.00 ± 0.00	83.06 ± 0.28	0.00 ± 0.00	0.00 ± 0.00	99.09 ± 0.13	36.94 ± 2.84	53.78 ± 2.99	0.99 ± 0.00	1h 26min
	DITTO (256)	33.16 ± 0.00	81.73 ± 0.00	47.18 ± 0.00	0.00 ± 0.00	83.66 ± 0.57	0.00 ± 0.00	0.00 ± 0.00	99.07 ± 0.30	38.06 ± 3.90	54.93 ± 4.02	0.99 ± 0.00	1h 20min
	DistilBERT (128)-15K	83.08 ± 4.54	77.48 ± 0.58	80.11 ± 1.92	0.01 ± 0.01	82.31 ± 1.44	0.02 ± 0.02	0.42 ± 0.12	98.06 ± 0.43	57.90 ± 9.17	**72.34 ± 7.47**	0.98 ± 0.00	**1h 15min**
	DistilBERT (128)-ALL	77.03 ± 3.83	79.46 ± 0.05	78.18 ± 2.00	0.00 ± 0.00	82.26 ± 0.78	0.00 ± 0.00	0.23 ± 0.05	98.76 ± 0.26	43.31 ± 5.10	60.03 ± 5.00	0.99 ± 0.00	1h 15min
	DistilBERT (128)-ALL-MEC	77.03 ± 3.83	79.46 ± 0.05	78.18 ± 2.00	0.00 ± 0.00	82.26 ± 0.78	0.00 ± 0.00	0.23 ± 0.05	98.57 ± 0.28	42.79 ± 4.91	59.50 ± 4.83	0.99 ± 0.00	1h 14min
	DistilBERT (128)-ALL ($\frac{1}{2}\gamma$)	77.03 ± 3.83	79.46 ± 0.05	78.18 ± 2.00	0.00 ± 0.00	82.26 ± 0.78	0.00 ± 0.00	0.23 ± 0.05	98.79 ± 0.25	43.23 ± 5.08	59.96 ± 4.97	0.99 ± 0.00	1h 15min
	DistilBERT (128)-ALL-BC	77.03 ± 3.83	79.46 ± 0.05	78.18 ± 2.00	0.00 ± 0.00	82.26 ± 0.78	0.00 ± 0.00	0.23 ± 0.05	98.76 ± 0.26	43.31 ± 5.10	60.03 ± 5.00	0.99 ± 0.00	1h 17min
Real Securities	DITTO (128)	19.96 ± 0.00	91.99 ± 0.00	32.80 ± 0.00	19.95 ± 0.01	92.10 ± 0.02	32.80 ± 0.02	0.20 ± 0.00	19.35 ± 0.46	17.59 ± 4.77	18.28 ± 2.84	0.19 ± 0.00	4.8 min
	DITTO (256)	19.96 ± 0.00	91.99 ± 0.00	32.80 ± 0.00	19.94 ± 0.00	92.11 ± 0.00	32.78 ± 0.00	0.20 ± 0.00	19.70 ± 0.01	20.93 ± 0.00	20.30 ± 0.01	0.19 ± 0.00	4.5 min
	DistilBERT (128)-ALL	99.76 ± 0.03	97.77 ± 0.01	98.76 ± 0.01	99.73 ± 0.02	98.08 ± 0.04	98.90 ± 0.01	1.00 ± 0.00	99.73 ± 0.02	98.00 ± 0.04	**98.86 ± 0.01**	1.00 ± 0.00	**2.6 min**
Synthetic Securities	DITTO (128)	97.26 ± 0.00	52.51 ± 0.00	68.20 ± 0.00	96.39 ± 0.10	54.58 ± 0.52	69.69 ± 0.45	0.98 ± 0.00	98.22 ± 0.22	44.88 ± 4.88	61.54 ± 4.65	0.99 ± 0.01	29.6 min
	DITTO (256)	97.26 ± 0.00	52.51 ± 0.00	68.20 ± 0.00	96.23 ± 0.25	57.08 ± 0.00	71.66 ± 0.06	0.98 ± 0.00	98.31 ± 0.33	56.68 ± 0.12	71.90 ± 0.18	0.99 ± 0.01	29.0 min
	DistilBERT (128)-15K	97.26 ± 0.00	57.06 ± 0.03	71.59 ± 0.04	96.05 ± 0.00	57.06 ± 0.03	71.59 ± 0.02	0.98 ± 0.00	98.08 ± 0.00	56.56 ± 0.04	71.71 ± 0.03	0.98 ± 0.00	**23.3 min**
	DistilBERT (128)-ALL	95.58 ± 2.03	53.28 ± 0.88	68.40 ± 0.20	87.81 ± 9.73	58.40 ± 1.50	69.82 ± 2.27	0.94 ± 0.04	96.70 ± 1.60	57.52 ± 1.02	**72.11 ± 0.34**	0.97 ± 0.01	23.4 min
WDC Products	DITTO (128)	19.71 ± 0.00	36.96 ± 0.00	25.71 ± 0.00	1.19 ± 0.31	50.38 ± 4.02	2.33 ± 0.59	0.01 ± 0.00	72.59 ± 2.21	9.02 ± 1.67	16.03 ± 2.69	0.84 ± 0.00	**31 sec**
	DITTO (256)	19.71 ± 0.00	36.96 ± 0.00	25.71 ± 0.00	20.34 ± 0.06	39.97 ± 0.86	26.96 ± 0.15	0.01 ± 0.00	74.14 ± 2.89	18.06 ± 2.72	28.96 ± 3.30	0.85 ± 0.04	32 sec
	DistilBERT (128)-ALL	39.64 ± 0.57	65.27 ± 1.15	49.32 ± 0.50	7.47 ± 4.78	71.4 ± 1.56	13.03 ± 7.75	0.43 ± 0.01	35.54 ± 1.29	57.93 ± 0.47	**44.04 ± 1.06**	0.53 ± 0.01	40 sec

in their names[14] which make false positive predictions more likely.

The *Post Graph Cleanup* scores show how false positive predictions can be detected and removed with graph-based techniques such as Algorithm 1. The recall scores are lower than the *Pre Graph Cleanup* ones because the GraLMatch Graph Cleanup also removes true positive matches, but the precision, F1 and Cluster Purity scores show that without the GraLMatch Graph Cleanup, a good entity group matching is not achieved.

Comparing the scores for the different models, we can see that DistilBERT(128)-ALL has a higher pairwise recall but lower precision than DistilBERT(128)-15K. This lower precision forces the GraLMatch Graph Cleanup to remove more predictions, which causes it to end up with a lower F1 score. DistilBERT(128)-15K has a lower recall due to being trained on fewer samples but achieves a higher precision. This higher precision means the GraLMatch Graph Cleanup does not need to remove as many predictions and makes it end up with the highest final F1 score.

Conversely, all DITTO variants consistently achieve a low pairwise precision score on both real and synthetic companies, leading to low *Pre Graph Cleanup* F1 and Cluster Purity scores. On the real companies, the GraLMatch Graph Cleanup works surprisingly well for DITTO and it achieves the highest F1 scores but this is not the case for synthetic companies. Given that the synthetic companies dataset is closer in size to the complete real companies dataset (see Table 2) and contains more challenging record groups to match (real companies mostly contain record groups obtained via identifiers which are easy to match), we come to the conclusion that having a high pairwise precision is the most determining factor in achieving a good entity group matching for large numbers of records.

[14]This is also the case in the complete real database, but the set of labeled records is considerably smaller in size and thus this phenomenon is not as significant.

DistilBERT(128)-15K is the model reaching the highest precision and the best *Post Graph Cleanup* F1 score for synthetic companies. This exemplifies the fact that training with more data (and thus investing more in labelling) and/or carrying out fine-tuning optimizations do not necessarily guarantee a good entity group matching performance, where precision is the most important factor.

Regarding the sensitivity of Algorithm 1, we can observe how DistilBERT (128)-ALL-BC obtains a lower *Post Graph Cleanup* Recall score than DistilBERT (128)-ALL, since the *Minimum Edge Cut* removes more true positive edges than Algorithm 1 albeit in a slightly shorter time. In turn, DistilBERT (128)-ALL-BC achieves the same scores as DistilBERT (128)-ALL with only a slightly longer running time because the *Edge Betweenness Centrality* selects the same edges for removal as the *Minimum Edge Cut*. DistilBERT (128)-ALL ($\frac{1}{2}\gamma$) obtains a half-way result between both previous setups, confirming our intuition that generally the *Minimum Edge Cut* is the faster but less accurate edge removal technique. The specific threshold values we choose prove to be a good compromise of accuracy vs speed although the small time differences signify that the running time of the experiment is dominated by the inference of the language models. Finally, the similar final Post Graph Cleanup F1 scores indicate that Algorithm 1 robustly achieves similar results with different choices of thresholds.

Regarding the groups affected by *data drift* events, record pairs presenting different textual attributes (names, descriptions) are likely being predicted as non-matches, since fine-tuning teaches the models to predict pairs based on text alignment. Consequently, scores largely represent how good models are at matching non-edge case groups. In a real setting, however, most edge case groups can be easily identified as they present matching identifiers but are predicted as non-matches by the language models. Their treatment will require additional knowledge sources, to identify the type of *data drift* event, and should be done on a case-by-case basis.

6.2.2 ***Securities Datasets.*** We now analyze the entity group matching results on the security datasets. Similar to the matching of companies, the pairwise matching performance (see first column of Table 4) is generally worse than the performance on the fine-tuning test pairs (see Table 3). We observe on the real security dataset that both DITTO models predominantly predict any pair to be a match, visible in the low precision while attaining a high recall, whereas DITTO (256) performed well on its test set. The synthetic dataset features more randomized identifiers and many pairs which cannot just be matched by finding an identifier overlap. Because of this, all models achieve precision above 95%, while their recall is around 53%, with the exception of DistilBERT(128)-15K at 57%.

During the pre graph cleanup (see second column of Table 4) we see a slight increase in recall at the cost of precision, though the securities are forming smaller clusters and thus do no exhibit as many transitively matched false positives. Because of this, the reduction in precision is much less severe, compared to the company datasets. Furthermore, we observe the same pattern as on the synthetic company dataset, where the DistilBERT(128)-15K variant trained on less data outperforms the DistilBERT(128)-ALL model.

After cleaning up the prediction graphs on the real security dataset, both DITTO models reach lower metrics compared to the pairwise matching performance with F1-Scores around 20%. This clearly shows how the initial predictions have failed to learn any useful pattern. The DistilBERT(128)-ALL model by comparison reaches a F1-Score of 98.86%, further indicating that the encoding structure of DITTO is not ideal for identifier-centric data. On the synthetic security dataset, the methods achieve a comparable precision, but the DITTO model needs to be double in size to match the DistilBERT(128)-15K model's performance in recall. DistilBERT(128)-ALL here reaches the highest F1-Score of 72.11%, though its precision is lower than the other models.

6.2.3 ***WDC Products.*** Finally, we analyze the entity group matching results on the WDC Products dataset, presented in the last rows of Table 4. As in the other datasets, the pairwise scores are worse than those obtained during fine-tuning due to the same reasons i.e. some true pairs being discarded by the blocking and the blocking candidate pairs being more difficult to classify than the test pairs evaluated during fine-tuning. In terms of entity group matching, we can again observe how the *Pre Graph Cleanup* Precision scores drop due to the effect of false positive pairwise predictions. The model with the highest pairwise precision, DistilBERT(128)-ALL, achieves the highest *Post Graph Cleanup* F1 score whereas DITTO (128) and DITTO (256), which present much lower pairwise precisions, both achieve lower F1 scores. Contrary to other datasets however, DITTO (256) achieves the best *Pre Graph Cleanup* F1 score but not the best *Post Graph Cleanup* F1 score. This is likely due to our Graph Cleanup method, which is not fit to matching settings where record groups of heterogeneous sizes have to be discovered such as the one this dataset presents. Using a better suited Graph Cleanup method should revert this phenomenon.

7 CONCLUSIONS

In this paper, we have discussed the particularities of the multi-source Entity Matching problem, that we refer to as *entity group matching*, where the challenge is to assign records, belonging to the same real-world entity, to the same group. We have presented two new benchmark datasets made up of companies and financial securities records, inspired by a real-world use-case. We have introduced the concept of *transitively matched records* and illustrated how false positive pairwise predictions affect the group assignment of large numbers of records leading to considerable numbers of false transitive matches. We have shown how these false pairwise predictions can largely be corrected via the use of graph-based algorithms such as the one we propose, *GraLMatch*.

Moreover, our experiments have shown that fine-tuning DistilBERT, a Transformer-based model with relatively few parameters, on a limited number of records yields a better out-of-sample performance than fine-tuning with more samples and/or with different fine-tuning optimizations. As our experiments illustrate, precision is the key to achieving a good entity group matching especially with large volumes of records.

Our approach *addresses and solves a pressing real-world problem in the financial industry.* Companies operating in this sector spend significant time on obtaining data from different data vendors to ensure comprehensive coverage across different instruments and geographies. The costs associated with these data licenses can reach millions of dollars for comprehensive data feeds from major providers like Bloomberg, Thomson Reuters (Refinitiv), etc. Our proposed approach will allow companies to have one-stop-shop access to financial data, which underpins almost every aspect of their operations.

ACKNOWLEDGMENTS

We thank Damian Tschirky and Gabriele Von Planta, both formerly of Move Digital AG, for their valuable insight during the course of the project. The work was funded by Innosuisse as an innovation project under the project number 54383.1 IP-ICT.

REFERENCES

[1] Ulrik Brandes. 2001. A faster algorithm for betweenness centrality. *Journal of mathematical sociology* 25, 2 (2001), 163–177.

[2] Ursin Brunner and Kurt Stockinger. 2020. Entity matching with transformer architectures - a step forward in data integration. *23rd International Conference on Extending Database Technology, Copenhagen, 30 March - 2 April 2020* (2020). https://doi.org/10.21256/ZHAW-19637

[3] Peter Christen and Dinusha Vatsalan. 2013. Flexible and extensible generation and corruption of personal data. *Proceedings of the 22nd ACM international conference on Information & Knowledge Management* (2013).

[4] Dong Deng, Raul Castro Fernandez, Ziawasch Abedjan, Sibo Wang, Michael Stonebraker, Ahmed K Elmagarmid, Ihab F Ilyas, Samuel Madden, Mourad Ouzzani, and Nan Tang. 2017. The Data Civilizer System.. In *Cidr*.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. http://arxiv.org/abs/1810.04805 cite arxiv:1810.04805Comment: 13 pages.

[6] Uwe Draisbach and Felix Naumann. 2010. DuDe: The Duplicate Detection Toolkit.

[7] Abdol-Hossein Esfahanian. 2013. Connectivity algorithms. *Topics in structural graph theory* (2013), 268–281.

[8] Alfio Ferrara, Stefano Montanelli, Jan Nößner, and Heiner Stuckenschmidt. 2011. Benchmarking Matching Applications on the Semantic Web. In *Extended Semantic Web Conference*.

[9] Anthony Gillioz, Jacky Casas, Elena Mugellini, and Omar Abou Khaled. 2020. Overview of the Transformer-based Models for NLP Tasks. In *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*. 179–183. https://doi.org/10.15439/2020F20

[10] Kai Hildebrandt, Fabian Panse, Niklas Wilcke, and Norbert Ritter. 2020. Large-Scale Data Pollution with Apache Spark. *IEEE Transactions on Big Data* 6 (2020), 396–411.

[11] Severin Holzer and Kurt Stockinger. 2022. Detecting errors in databases with bidirectional recurrent neural networks. In *25th International Conference on Extending Database Technology, Edinburgh (online), 29 March-1 April 2022*. OpenProceedings, 364–367.

[12] Ekaterini Ioannou, Nataliya Rassadko, and Yannis Velegrakis. 2013. On Generating Benchmark Data for Entity Matching. *Journal on Data Semantics* 2, 1 (01 Mar 2013), 37–56. https://doi.org/10.1007/s13740-012-0015-8

[13] Pradap Konda, Sanjib Das, Paul Suganthan G. C., AnHai Doan, Adel Ardalan, Jeffrey R. Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeff Naughton, Shishir Prasad, Ganesh Krishnan, Rohit Deep, and Vijay Raghavendra. 2016. Magellan: Toward Building Entity Matching Management Systems. *Proc. VLDB Endow.* 9, 12 (aug 2016), 1197–1208. https://doi.org/10.14778/2994509.2994535

[14] Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2010. Evaluation of entity resolution approaches on real-world match problems. *Proceedings of the VLDB Endowment* 3 (2010), 484 – 493.

[15] Yuliang Li, Jinfeng Li, Yoshi Suhara, An Hai Doan, and Wang Chiew Tan. 2023. Effective entity matching with transformers. *VLDB Journal* (1 2023), 1–21. https://doi.org/10.1007/S00778-023-00779-Z/TABLES/14

[16] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv e-prints*, Article arXiv:1907.11692 (July 2019), arXiv:1907.11692 pages. https://doi.org/10.48550/arXiv.1907.11692 arXiv:cs.CL/1907.11692

[17] Zhengjie Miao, Yuliang Li, and Xiaolan Wang. 2021. Rotom: A meta-learned data augmentation framework for entity matching, data cleaning, text classification, and beyond. In *Proceedings of the 2021 International Conference on Management of Data*. 1303–1316.

[18] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep Learning for Entity Matching: A Design Space Exploration. *Proceedings of the 2018 International Conference on Management of Data* (2018).

[19] Avanika Narayan, Ines Chami, Laurel Orr, Simran Arora, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data? *arXiv e-prints*, Article arXiv:2205.09911 (May 2022), arXiv:2205.09911 pages. https://doi.org/10.48550/arXiv.2205.09911 arXiv:cs.LG/2205.09911

[20] George Papadakis, George Mandilaras, Luca Gagliardelli, Giovanni Simonini, Emmanouil Thanos, George Giannakopoulos, Sonia Bergamaschi, Themis Palpanas, and Manolis Koubarakis. 2020. Three-dimensional Entity Resolution with JedAI. *Information Systems* 93 (05 2020), 101565. https://doi.org/10.1016/j.is.2020.101565

[21] Ralph Peeters and Christian Bizer. 2021. Dual-Objective Fine-Tuning of BERT for Entity Matching. *Proc. VLDB Endow.* 14, 10 (jun 2021), 1913–1921. https://doi.org/10.14778/3467861.3467878

[22] Ralph Peeters and Christian Bizer. 2022. Supervised Contrastive Learning for Product Matching. *Companion Proceedings of the Web Conference 2022* (2022).

[23] Ralph Peeters and Christian Bizer. 2023. Using ChatGPT for Entity Matching. *arXiv e-prints*, Article arXiv:2305.03423 (May 2023), arXiv:2305.03423 pages. https://doi.org/10.48550/arXiv.2305.03423 arXiv:cs.CL/2305.03423

[24] Ralph Peeters, Reng Chiz Der, and Christian Bizer. 2024. WDC Products: A Multi-Dimensional Entity Matching Benchmark. *Proceedings of EDBT* (2024).

[25] Anna Primpeli and Christian Bizer. 2021. Graph-Boosted Active Learning for Multi-Source Entity Resolution. In *The Semantic Web – ISWC 2021: 20th International Semantic Web Conference, ISWC 2021, Virtual Event, October 24–28, 2021, Proceedings*. Springer-Verlag, Berlin, Heidelberg, 182–199. https://doi.org/10.1007/978-3-030-88361-4_11

[26] Anna Primpeli and Christian Bizer. 2022. Impact of the Characteristics of Multi-source Entity Matching Tasks on the Performance of Active Learning Methods. In *Extended Semantic Web Conference*.

[27] David Reinsel, John Gantz, and John Rydning. 2017. Data age 2025: The evolution of data to life-critical. *Don't Focus on Big Data* 2 (2017).

[28] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[29] Tzanina Saveta, Evangelia Daskalaki, Giorgos Flouris, Irini Fundulaki, Melanie Herschel, and Axel-Cyrille Ngonga Ngomo. 2015. LANCE: Piercing to the Heart of Instance Matching Tools. In *International Workshop on the Semantic Web*.

[30] Saravanan Thirumuruganathan, Han Li, Nan Tang, Mourad Ouzzani, Yash Govind, Derek Paulsen, Glenn Fung, and AnHai Doan. 2021. Deep Learning for Blocking in Entity Matching: A Design Space Exploration. *Proc. VLDB Endow.* 14, 11 (jul 2021), 2459–2472. https://doi.org/10.14778/3476249.3476294

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv e-prints*, Article arXiv:1706.03762 (June 2017), arXiv:1706.03762 pages. https://doi.org/10.48550/arXiv.1706.03762 arXiv:cs.CL/1706.03762

[32] Jin Wang, Yuliang Li, and Wataru Hirota. 2021. Machamp: A Generalized Entity Matching Benchmark. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (2021).

[33] Runhui Wang, Yuliang Li, and Jin Wang. 2022. Sudowoodo: Contrastive Self-supervised Learning for Multi-purpose Data Integration and Preparation. arXiv:cs.DB/2207.04122

[34] Dezhong Yao, Yuhong Gu, Gao Cong, Hai Jin, and Xinqiao Lv. 2022. Entity Resolution with Hierarchical Graph Attention Networks. *Proceedings of the 2022 International Conference on Management of Data* (2022).