Record Linkage Using A Domain Knowledge Ruleset

Dongyun Nie and Mark Roantree

Abstract

Application areas such as healthcare and insurance see many patients or clients with their lifetime record spread across the databases of different providers. Record linkage is the task where algorithms are used to identify the same individual contained in different datasets. In cases where unique identifiers are found, linking those records is a trivial task. However, there are very high numbers of individuals who cannot be matched as common identifiers do not exist across datasets and their identifying information is not exact or often, quite different (e.g. a change of address). In this research, we provide a new approach to record linkage which also includes the ability to detect relationships between customers (e.g. family). A validation is presented which highlights the best parameter and configuration settings for the types of relationship links that are required.

Record Linkage Using A Domain Knowledge Ruleset*

Dongyun Nie and Mark Roantree
Insight Centre for Data Analytics
School of Computing, Dublin City University, Ireland
dongyun.nie@insight-centre.org, mark.roantree@dcu.ie

Keywords: Record Linkage; Relationships; Customer Knowledge

Abstract: Application areas such as healthcare and insurance see many patients or clients with their lifetime record spread across the databases of different providers. Record linkage is the task where algorithms are used to identify the same individual contained in different datasets. In cases where unique identifiers are found, linking those records is a trivial task. However, there are very high numbers of individuals who cannot be matched as common identifiers do not exist across datasets and their identifying information is not exact or often, quite different (e.g. a change of address). In this research, we provide a new approach to record linkage which also includes the ability to detect relationships between customers (e.g. family). A validation is presented which highlights the best parameter and configuration settings for the types of relationship links that are required.

1 INTRODUCTION

Customer Relationship Management (CRM) allows companies to manage their interactions with current and potential customers. CRM combines people, processes and technology to try to understand a customer's needs and behaviour. Getting to *know* each customer using data mining techniques and by adopting a customer-centric business strategy helps the organization to be proactive, offering more products and services for improved customer retention and loyalty over longer periods of time (Chen and Popovich, 2003). By using data analysis on customer history, the goal is to improve business relationships with customers, specifically focusing on customer retention and ultimately improving sales growth. A metric known as *Customer Lifetime Value* (CLV) can be regarded as a sub-topic of CRM which focuses on predicting the net profit that can accrue from the future relationship with a customer (Di Benedetto and Kim, 2016).

As CLV calculations require predictive algorithms to classify new customers, the algorithm's ability to accurately make these predictions depends on the datasets that generate the predictive models. For our research, this requires the generation of a full customer profile, in effect, combining all records from across multiple enterprise systems to generate the complete customer record. This type of integration process is known as record linkage (Bilenko and Mooney, 2003), (Bilenko et al., 2006) where there is no single key to integrate separate customer histories and instead, a form of fuzzy match is required to detect where the same customer has been found. The research presented in this paper describes our work to tackle this particular problem for insurance data, policies and contracts, together with an industry partner.

Our work is with a large insurance company based in Ireland. Their issue is they are unaware when they are dealing with the same customer or a relative of that customer, due to the fact that the customer may interface with the company in different ways: through agents, online or through an offer from a third party. Thus, there is no certainty of a link between these policies or accounts. In this collaboration, there is a requirement to develop an integration strategy to build the complete historical customer record. Once constructed, this dataset can be used as part of the validation process, a process which is overlooked in the CLV literature. A final motivation for creating this dataset is to identify new useful relationships such as family members who are customers in their own right.

Tasks for data integration include data preparation (Pyle, 1999), knowledge fusion (Dong et al., 2014) in addition to matching the data (Bhattacharya and Getoor, 2007; Cohen et al., 2003; Rahm, 2016; Yujian and Bo, 2007; Etienne et al., 2016; Ferguson et al., 2018). Knowledge fusion is an information integra-

*Research funded by Science Foundation Ireland under grant number SFI/12/RC/2289

tion process which merges information from repositories to construct knowledge bases. Traditionally, the knowledge base is built using existing repositories of structured knowledge. Record linkage is a specific problem within integration which has a unique computation problem. Matching all records in a pairwise fashion requires 499,500 comparisons for just 1,000 records and 4,999,950,000 comparisons for 100,000 records. This presents a significant challenge as the size of the dataset increases. Early attempts to address this problem (Baxter et al., 2003) included blocking where the matching space could be significantly reduced by splitting data into a large number of segments. By introducing blocking predicates (Bilenko et al., 2006), this technique was improved to exploit domain semantics for improved segmentation. However, most of these efforts used synthetic datasets e.g. (Bilenko et al., 2006) or healthcare records e.g. (Mamun et al., 2016). While trying to use these techniques in a very specific domain - insurance datasets - we encountered issues with a higher number of lost matches. Furthermore, we had a specific task of matching clients with family members, an approach not discussed in current related research.

1.1 Contribution

The construction of a unified record for all customers requires a fuzzy matching strategy, usually relying on the construction of a similarity matrix across all customers. However, this has two major challenges: the construction and evolution costs of a similarity matrix are prohibitive and initial experiments showed that a single similarity value across many attributes had poor results in terms of matching accuracy. In this work, we present a customer matching approach which uses a *modified* form of Agglomerative Hierarchical Clustering (AHC) that incorporates a method for overlapping segments. This hybrid approach of data mining, together with a companion ruleset to detect and link: components of the same customer record, clients with family members; and clients with co-habitants who have also bought policies, allows relatively fast matching while achieving high levels of accuracy. An evaluation is provided to illustrate the levels of matching that were achieved and a human assisted validation process.

1.2 Paper Structure

The remainder of this paper is structured as follows: in §2, we present a review of related research in this area; in §3, we present an overview of the system and the methodology that we used to integrate data for constructing unified client records; in §4, we introduce our segmentation method. in §5, we specify the detail of matching using modified Agglomerative Hierarchical Clustering; in §6 we present our experiments and an evaluation in terms of high level user group queries; and finally, §7 contains our conclusions.

2 RELATED RESEARCH

To integrate large amounts of source data, the authors in (Rahm, 2016) developed an approach to integrate schemas, ontologies and entities. Their purpose was to provide an approach that could match large numbers of data sources not only for pairwise matching but also for holistic data integration through many data sources. For a complex schema integration, they first used an intermediate result to merge schemas until all source schemas have been integrated. For entity integration, they first clustered data by semantic type and class, where only entities in one cluster were compared with each other. However, when clustering very large datasets, the time consumption increases rapidly. This is a well known problem and, in our work, we have the same issue. Their approach cannot be copied in our research as they use Linked Open Data while insurance data does not have the same properties as Linked Data. Furthermore, our unified record must create a relationship graph (connected families and co-habitants) between every customer record. Thus, if we adopt their approach, a further layer of processing is still required.

In (Bhattacharya and Getoor, 2007), the authors proposed a clustering algorithm that uses both attributes and relational information to determine the actual join while having an efficient implementation. They evaluated a series of entity resolution approaches: attribute-based entity resolution, naive relational entity resolution, and collective relational entity resolution. Additionally, multiple similarity measures were applied: common neighbours, Jaccard coefficient, Adamic/Adar similarity, Adar similarity with ambiguity estimate, higher-order neighborhoods, and negative constrains from relationships. They demonstrated while investigating a real world bibliographic dataset, their similarity measures ambiguity (corresponding to the disambiguation aspect of resolution) and dispersion (corresponding to the identification aspect) significantly outperformed other algorithms. Their work provides a solid understanding of how similarity measures work with real-world data. However, the size of their experiment dataset is thousands of reference records. Enterprise recordsets such as that used in our research are far bigger. As a result, the time required for calculating the similarity between pairwise record is $O(n^2)$. While we may be able to adopt some elements of this research, we will be required to manage high data volumes.

If we wish to focus specifically on designing a matching and integration solution for large recordsets, it is necessary to efficiently check the similarity between every pair of customer records, which is further complicated by string attributes. In (Mamun et al., 2016), the author used similar blocking techniques for segmenting and, like our approach, allows duplicates. They then match using a minimum threshold distance between two clusters to remove duplicates. However, where the attributes and size of the clusters increase, the calculation time increases exponentially. In our work, we must match multiple relationships where, for a very low threshold, it creates very large segments. While they employ a post-processing step to avoid merging all records into a single large cluster, it does not allow the detection of different types of relationships.

The authors in (McCallum et al., 2000) present similar research to ours where they employ two steps to match references. Firstly, they used a method called Canopies, which offered a *quick and dirty* text distance matrix to find the relevant data within a threshold and put them in subsets. The fast distance matrix is to calculate the distance using an inverted index, which calculates the number of common words in a pairwise reference. A threshold will be applied to determine subsets and, similar to our approach, subsets may overlap. They then use greedy agglomerate clustering to identify pairs of items inside Canopies. While there are similarities in our two approaches, essentially there are limiting their approach to matching author names to detect the same author. Our matching is multi-dimensional, with similarity matrices across 9 attributes, and we are seeking to detect 3 forms of relationships, and not simply the *author-author* relationship in this work.

Many researchers like (Huang, 1998; Larsen and Aone, 1999; Hotho et al., 2003; Sedding and Kazakov, 2004; Bilenko et al., 2006; Mamun et al., 2016; Ferguson et al., 2018) all provide methods for managing text values while clustering where the common method is to use blocking techniques with n-grams or k-mer and convert strings to vectors. One applies tf (term frequency) or *tf-idf* (term frequency by inverse document frequency) to weight the vector, so that clustering vectors calculate the distance using similarities. All of these experiments use either semantic datasets, reference datasets or text documents. In attempting to use these approaches, we are faced with many mismatches as records for the same customer (or for family members) were placed in separate segments. However, string matching approaches as discussed in this literature are often inadequate where we are trying to determine if two entities (customers) are the same. The nature of string matching will give many false positives (for actual customers) and can miss - or rank much lower - two entities which may refer to the same customer. Consider a situation where the customer ”Ryan” is misspelled as ”Ryen”. This is the same customer but may get a lower score than ”Ryan” and ”O Ryan” which *are* different customers. These algorithms will require a level of adaptation for usage in entity resolution.

3 MATCHING METHODOLOGY

Our methodology comprises 5 steps: pre-processing; segmenting the recordset; application of the matching algorithm; using a ruleset to improve matching results; and validation, with an overview of our system architecture shown in Fig 1.

Step 1: Pre-processing This step involves cleaning data before matching can commence. Firstly, all characters are converted to lowercase to eliminate the dissimilarity due to case sensitivity. Secondly, all non-alpha-numeric characters are removed. Finally, the 4-attribute address is concatenated but the most abstract level of granularity (normally country) is removed.

Step 2: Dataset Segmentation Our validation dataset contains 194,396 records and will require approximately 20 billion comparison operations for a single evaluation using a single attribute. For this reason, the first task is to segment the recordset with the goal of minimizing the possibility of a customer having records in separate segments, as those records will never be matched. The most commonly used segmentation methods are clustering with vectoring attributes (Baxter et al., 2003). However, in almost all of these research projects, they seek only to match the same *person* which is referred to as *Client-Client* matching using our approach. However, a separate goal is to link family members and non-family member cohabitants. Details are provided in §4.

Step 3: Clustering Client Records. We adopt a clustering approach based on Agglomerative Hierarchical Clustering (AHC) (Day and Edelsbrunner, 1984), where a similarity matrix is computed to represent the distance between each pair of records. For us, the process stops after a deliberately low threshold for distance (dissimilarity) has been reached. Our first point of difference with traditional AHC lies in our

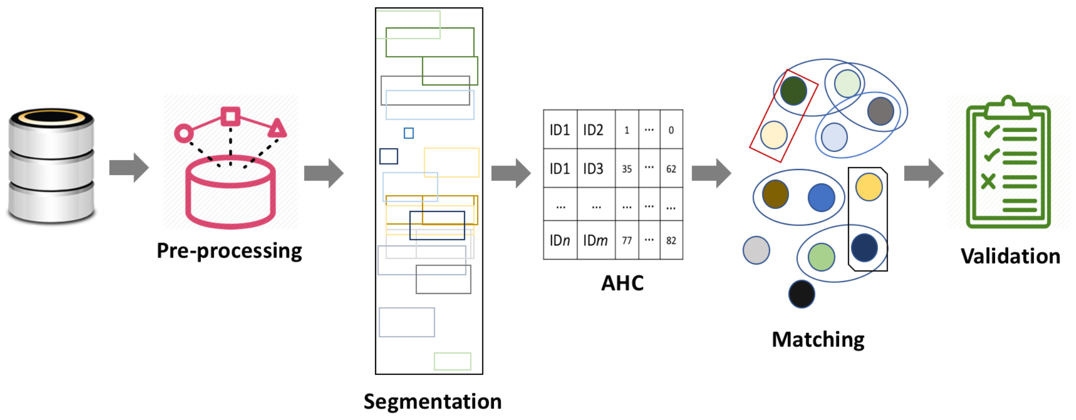


Figure 1: Pre-processing, Segment, Comparing and Matching Client Records

construction of the similarity matrix. We do not construct a single 2-dimensional matrix but instead compute a multidimensional matrix which enables us to examine distance measures across different variables. We chose this method due to poor results obtained when using a single aggregated distance measure across all variables. There are nine dimensions in our current similarity matrix as presented in Table 1, with each dimension (matrix) given a specific label comprising *SM_* and the name of the attribute. This reference to similarity dimensions (or matrices) is also used in the rules presented in §5. The *SM_BirthDate* dimension captures the distance between Dates of Birth; *SM_FirstName* and *SM_LastName* for the first and last names; *SM_Address* for the distance between address strings; *SM_Email*, *SM_Mobile*, *SM_HomePhone*, *SM_WorkPhone* and *SM_Fax* for the distance between each type of contact details.

Table 1: Similarity Matrix usage in Relationship Matching

Ref	Similarity Matrix	Client	Family	CoHab
1	*SM_BirthDate*	Y	N	N
2	*SM_FirstName*	Y	N	N
3	*SM_LastName*	Y	Y	N
4	*SM_Address*	O	O	Y
5	*SM_Email*	O	O	N
6	*SM_Mobile*	O	O	N
7	*SM_HomePhone*	O	O	N
8	*SM_WorkPhone*	O	O	N
9	*SM_Fax*	O	O	N

Step 4: Application of Rules. While using a multidimensional similarity matrix allows for a more fine grained comparison of distance between client records, the application of all dimensions was not suited in all matching requirements. Furthermore, we required a facility to apply different thresholds across the dimensions. There are three types of matches required in our research: client matches (records for the same client); family matches (family members for a client); and domiciled (where non family members reside at the same address).

To count the number of matches for family, it is necessary to exclude same-client matches and for the domiciled matches is necessary to exclude family and same-client matches.

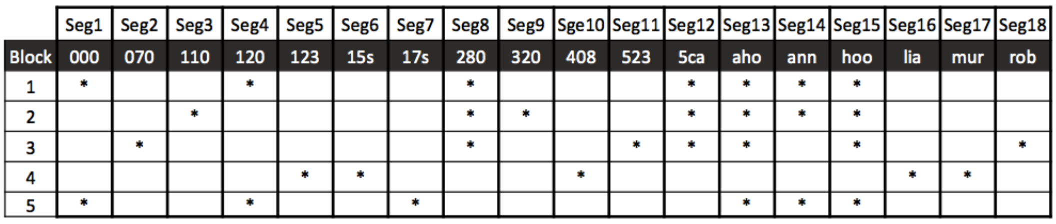
4 DATASET SEGMENTATION

Similar to other approaches, we seek to match two different records for the same client. However, we must also identify family members as a parent or spouse may buy a policy for their child or partner. It is not unusual for this type of relationship to have a higher matching score than for two records the same client. Our approach also matches (non family member) co-habitants. In this section, we present a hybrid segmentation method which seeks to reduce the matching (search) space between records.

While attempting record linkage for a large dataset, most approaches (e.g. (Etienne et al., 2016; Ferguson et al., 2018)) to segmentation adopt a clustering approach that employs blocking and a form of vectorization for fast processing of the large pairwise matching required in their similarity matrix. Blocking involves the selection of a block (always small e.g. 3 chars) of consecutive characters which are used for distance matching. This can be illustrated using Table 2 which contains 5 sample records after our pre-processing step. Customer records 1 and 5 refer to the same client where a mistake was made for dimension *BirthDate*. Customer records 1, 2 and 3 are family members with shared Contact (Dimension 5 to

Table 2: Sample Records

Record	*BirthDate*	*FirstName*	*LastName*	*Address*	*Email*	*Mobile*	*HomePhone*	*WorkPhone*	*Fax*
1	12091990	anna	hood	5capelst	ahood21gmailcom	0876720000	013333280	null	null
2	11051964	ann	hood	5capelst	ahood21gmailcom	0860802320	013333280	null	null
3	07041993	robert	hood	5capelst	ahood21gmailcom	0897034523	013333280	null	null
4	12301992	liam	murphy	15silloguerdballymun	liam2murphygmailcom	0867723408	null	null	null
5	12071990	anna	hood	17silloguer oadballymun	annahood1gmailcom	353876720000	null	null	null



	Seg1	Seg2	Seg3	Seg4	Seg5	Seg6	Seg7	Seg8	Seg9	Sge10	Seg11	Seg12	Seg13	Seg14	Seg15	Seg16	Seg17	Seg18
Block	000	070	110	120	123	15s	17s	280	320	408	523	5ca	aho	ann	hoo	lia	mur	rob
1	*			*				*				*	*	*	*			
2			*					*	*			*	*	*	*			
3		*						*			*	*	*		*			*
4					*	*				*						*	*	
5	*			*			*						*	*	*			

Figure 2: Segmentation by Blocking using Table 2

9) information. Additionally, customer 4 lives with customer 5. Figure 2 allocates the sample records from Table 2 into their respective segments (one of 18 possible segments) based on the block that represents each segment. Our *overlapping* approach is different to other approaches: if that block is found in any attribute in the same record, it is placed into that segment. Thus, a record can appear in more than one segment, e.g. Record #1 is placed into segments 1, 4, 8, 12, 13, 14 and 15.

This was necessary as, in early tests using the DBSCAN clustering method (Han et al., 2011), up to 30% of records for the same clients were in separate segments, meaning they could never be matched. On the other end of the scale, setting the distance to 13, all records were placed in the same cluster, meaning the number of matching operations was too large to compute.

In (Etienne et al., 2016), the authors employed prefix blocking for attributes *FirstName* and *LastName* and other approaches included blocking for *Address*, *BirthDate* and *Email*. Essentially, this meant taking a block of n-characters from the start of each string for comparison purposes. For contact attributes, we employ suffix blocking. This meant taking a block of characters from the end of each string for attributes (*Mobile*, *HomePhone*, *WorkPhone* and *Fax*). This had the advantage of avoiding issues with country and area codes where they may or may not exist. The way (prefix or suffix) of blocking is consistent for all experiments in §6.

After segmentation, the matching algorithm presented in §5 is employed to deliver record linkage across the entire dataset.

5 RULE ASSISTED MATCHING

In constructing similarity matrices, we treat all attributes as strings and generate Levenshtein distance (Kruskal, 1983) measures. The end goal is a unified customer record containing three different relationships: records for the same client; records of family members; and those of cohabitants (domicile). The workflow and various concepts are shown in Fig-3. We begin by constructing the multi-dimensional similarity matrix by applying a similarity measure to each attribute. We then apply the rules which set distance thresholds for matching purposes to cluster according to the different relationships. Finally, we merge the related records into unified client records.

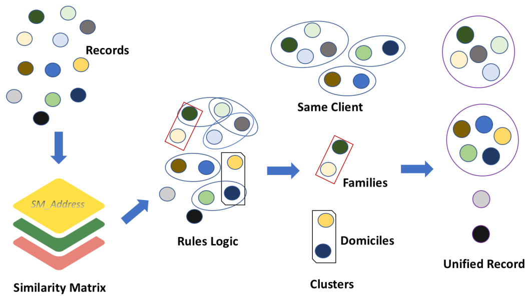


Figure 3: Matching Using Modified AHC

While Null values are very common in a real-world customer dataset, it makes it even more difficult to deal with the problem of evaluating similarity between two customers. If a value of Null is present for the same attribute in both records, the distance will be 0, meaning that providing no information will result in an exact match. Thus, Null values distort our methodology and thus, we punish Null values during construction of the similarity matrix. This was initially managed in two ways: using the *average* distance or the *maximum* distance value for this attribute which is similar to single link and complete link calculations (Day and Edelsbrunner, 1984). However, we are using a multidimensional similarity matrix. The results of experiments showed that any number greater than the maximum distance threshold we apply will fulfill the requirement of punishing the null value. In our case, we assigned 6 to a similarity matrix if this attribute in both records are null during the construction process.

We have 3 categories of rules, Client, Family or Domicile, which are applied according to the type of match required. The size of the similarity matrices depend on the length of the block and the number of the attributes. We describe the different configurations of blocking and attributes in section 6.

5.1 Client Rule

Definition 1. *Client-Client_Rule*
$[DOB\_Check]$ *and*
$([Full\_Name\_Check]^{*}\ or\ )$ *and*
$[Contact\_Detail\_Check]$

In Definition 1, we introduce the *Client-Client_Rule* as a rule which must have 3 separate clauses, each separated by a logical *and* operator. All conditions must evaluate to true if records are to be clustered (matched). The condition $([Full\_Name\_Check]^{*}\ or\ )$ will contain one or more than one clause of $Full\_Name\_Check$ separated by a logical *or* operator.

Definition 2. *DOB_Check*
$SM\_BirthDate[i,j] \leq T_{DOB}$

In Definition 2, the *DOB_Check* clause is specified as a Boolean statement. In this case, the similarity for records *i* and *j* are tested using the *SM_BirthDate* similarity matrix against a specified threshold value T_{DOB}.

Definition 3. *Full_Name_Check*
$[FirstName\_Check]$ *and*
$[LastName\_Check]$

In Definition 3, the $Full\_Name\_Check$ clause is a Boolean statement with two conditions $FirstName\_Check$ and $LastName\_Check$ separated by a logical *and* operator.

Definition 4. *FirstName_Check*
$SM\_FirstName[i,j] \leq T_{FName}$

In Definition 4, the *FirstName_Check* clause is specified as a Boolean statement. In this case, the similarity for records *i* and *j* are tested using the *SM_FirstName* similarity matrix against a specified threshold value T_{FName}.

Definition 5. *LastName_Check*
$SM\_LastName[i,j] \leq T_{LName}$

In Definition 5, the *LastName_Check* clause is specified as a Boolean statement. In this case, the similarity for records *i* and *j* are tested using the *SM_LastName* similarity matrix against a specified threshold value T_{LName}.

There are two similarity matrices *SM_FirstName* and *SM_LastName* along with their specified threshold values T_{FName} and T_{LName} in $Full\_Name\_Check$ clauses need to be tested together. The threshold applied to this clause may present in multiple ways such that the sum of T_{FName} and T_{LName} is equal to the given number.

For example, if the threshold applied to $Full\_Name\_Check$ is 2, the possible combination of a sub-clause sum of 2 is presented in Example 1.

Example 1.
(SM_FirstName[i,j] $\leq$ *0 and SM_LastName[i,j]* $\leq$ *2) or (SM_FirstName[i,j]* $\leq$ *1 and SM_LastName[i,j]* $\leq$ *1) or SM_FirstName[i,j]* $\leq$ *2 and SM_LastName[i,j]* $\leq$ *0)*

Definition 6. *Contact_Details_Check*
$[Address\_Check]$ *or*
$[Contact\_Check]$

In Definition 6, the *Contact_Details_Check* clause is a Boolean statement with two clauses $Address\_Check$ and $Contact\_Check$ separated by a logical *or* operator.

Definition 7. *Address_Check*
$SM\_Address[i,j] \leq T_{AD}$

In Definition 7, the $Address\_Check$ clause is specified as a Boolean statement. In this case, the similarity for records *i* and *j* are tested using the *SM_Address* similarity matrix against a specified threshold value T_{AD}.

Definition 8. *Contact_Check*
$SM\_Email[i,j] \leq T_{EM}$ *or*
$SM\_Mobile[i,j] \leq T_{MO}$ *or*
$SM\_HomePhone[i,j] \leq T_{HP}$ *or*
$SM\_WorkPhone[i,j] \leq T_{WP}$ *or*

$SM\_Fax[i,j] \leq T_{Fax}$

The *Contact_Check* checks the similarity for records i and j against a list of contact similarity metrics: *SM_Email*; *SM_Mobile*; *SM_HomePhone*; *SM_WorkPhone*; *SM_Fax*. Each similarity matrix had its assigned threshold T_{EM} for *SM_Email*; T_{MO} for *SM_Mobile*; T_{HP} for *SM_HomePhone*; T_{WP} for *SM_WorkPhone* and T_{Fax} for *SM_Fax*.

An example of the *Client_Rule* setting the threshold for each clause to 3 is shown in Example 2:

Example 2.
(SM_BirthDate[i,j] $\leq$ 3) and
((SM_FirstName[i,j]$\leq$ 0 and SM_LastName[i,j]$\leq$ 3) or
(SM_FirstName[i,j]$\leq$ 1 and SM_LastName[i,j]$\leq$ 2) or
(SM_FirstName[i,j]$\leq$ 2 and SM_LastName[i,j]$\leq$ 1) or
(SM_FirstName[i,j]$\leq$3 and SM_LastName[i,j]$\leq$ 0)) and
(SM_Address[i,j] $\leq$ 3 or SM_Email[i,j] $\leq$ 3 or
SM_Mobile[i,j] $\leq$ 3 or SM_HomePhone[i,j] $\leq$3 or
SM_WorkPhone[i,j] $\leq$ 3 or SM_Fax[i,j] $\leq$3)

5.2 Family Rules

Definition 9. *Client-Family_Rule*
$[LastName\_Check]$ *and*
$[Contact\_Detail\_Check]$

In Definition 9, we introduce the *Family_Rule* as a rule which must have two separate clauses, each separated by a logical *and* operator. In this rule, the two clauses included are Definition 5 and Definition 8.

An example of the *Family_Rule* setting the threshold for each clause to 3 is shown in Example 3:

Example 3.
(SM_LastName[i,j]$\leq$ 3) and
(SM_Address[i,j] $\leq$ 3 or SM_Email[i,j] $\leq$ 3 or
SM_Mobile[i,j] $\leq$ 3 or SM_HomePhone[i,j] $\leq$3 or
SM_WorkPhone[i,j] $\leq$ 3 or SM_Fax[i,j] $\leq$3)

5.3 Domicile Rules

Definition 10. *Client-Domicile_Rule*
$[Address\_Check]$

In Definition 10, we introduce the *Domicile_Rule* as a rule which tests the domiciled clients using clause Definition 7.

An example of the *Domicile_Rule* assigning the threshold to 3 is shown in Example 4:

Example 4.
SM_Address[i,j] $\leq$ 3

6 EVALUATION

In order to provide as in-depth a validation as possible, we ran 3 different sets of experiments, with different configurations and thresholds. Experiment 1 used all of the similarity matrices presented in Table 1. Exp1.1 used a blocking method with a length of 3 for *BirthDate* (DOB), *FirstName* (FN), *LastName* (LN); for all contact details - *Address*, *Email*, *Mobile*, *HomePhone*, *WorkPhone* and *Fax* - the length of blocking is 6. Experiment 2 used similarity matrices 3 to 9 (a combination of last name and all contact details) from Table 1, using 2 different blocking configurations. Experiment 2.1 used a blocking of length of 3 for *LastName* and length of 5 for all contact details while experiment 2.2 used a length of 5 for all contact details. Finally, experiment 3 used similarity matrices 4-9 (contact details only) with 3 different blocking configurations. In experiment 3.1, the length is 4; in experiment 3.2, the length is 5; and finally, in experiment 3.3, the length of blocks is 6.

6.1 Results

We use 2 tables to present our results: Table 3 presents the configuration details for each of 6 experiments while Table 4 presents the total matches and accuracy for different thresholds across all 6 experiments.

The first column in Table 3, *Exp*, is the label for the 3 sets of experiments, each with different configurations for the block length (*Block Length*). Columns 2-5 show the length assigned to the attributes: the second column represents the length for attribute *BirthDate* (*DOB*); *FN* and *LN* columns for *FirstName* and *LastName*; the Contact column represents all the contact details including *Address*, *Email*, *Mobile*, *HomePhone*, *WorkPhone* and *Fax*. A value between 3 and 6 indicates the block length for strings and *n/a* indicates that this attribute was not used in the experiment. The *Dims* column lists the number of similarity dimensions used in the matching process. *Records* refers to the size of the recordset involved in that experiment with the total number of segments created listed in the *Segment* column. The total number of records compared for a single dimension of the similarity matrix are shown in *Comparison* and finally, the number of records in the largest segment is shown in *Max*.

The goal of our research is to achieve the maximum number of matches while identifying any limitations caused by threshold values for each rule. Thus, our evaluation is focused on measuring matching accuracy, as validated by our industry partner. In certain cases, they require very high levels of accuracy while

Table 3: Experiment Configurations and Matching Requirements

Exp	Block Length				Dims	Records	Segment	Comparison	Max
	DOB	FN	LN	Contact					
1.1	3	3	3	6	9	1,168,406	311,150	843,109,791	17,292
2.1	n/a	n/a	3	5	7	808,396	137,177	185,526,138	8,026
2.2	n/a	n/a	3	6	7	778,666	271,215	137,298,018	4,469
3.1	n/a	n/a	n/a	4	6	583,776	42,016	186,793,296	13,090
3.2	n/a	n/a	n/a	5	6	583,924	141,549	94,824,563	8,026
3.3	n/a	n/a	n/a	6	6	584,290	268,451	46,904,079	4,455

in other cases, they are happy with a reduced level if we can provide far higher numbers of matches. The results in Table 3 show that decreasing the length during blocking will decrease the number of segments created but an increase in segment size will see an increase in the number of comparisons required. Experiment running time is dependent on the number of comparisons in each experiment.

For all 6 experimental configurations, we ran 4 client matching experiments, 3 client-family experiments and 1 experiment for co-habitants, as shown in Table 4. Rows 2 to 5 (labelled with rules CC0, CC1, CC2 and CC3) in Table 4 show the results of matching by the *Client-Client_Rule* with threshold values from 0 to 3 for every clause. Rows labelled CF0, CF1 and CF2 show the result of the *Client-Family_Rule* with threshold values of 0, 1 and 2 respectively for the two clauses in this rule. The last row CD0 is the result for *Client-Domicile_Rule*, always with a threshold value set to 0. The last row *Total* represents the total number of matches for all matching experiments (sum of CC3, CF2, and CDO) within the listed *Exp*. The Accuracy (*Acc %*) for the total is the true accurate matches in all matching experiments divided by the Total.

- As expected, applying a very low threshold (distance value) will result in very high accuracy. Increasing the threshold will match more records but will, as a result, reduce the accuracy. In general terms, the number of matches increases, row by row, within each matching category.
- A higher distance threshold also captures those matches found using a lesser threshold. For example, the 35,856 matches detected in Exp2.2 using threshold CF1 includes the 30,330 matches for CF0 together with the additional 5,526 detected using the higher distance value of CF1.
- For all blocking experiments (1.1 to 3.3), where the threshold is set to 0 (CC0, CF0 and CD0), identical records are matched and thus, the same level of accuracy is achieved.
- Setting the threshold to zero (CC0, CF0 and CD0) will override all experimental configurations: neither blocking algorithms nor matrix usage has any effect.
- If we look across the experiments, when the matching criteria is more strict (reduction in attribute comparisons), matches decrease, with the accuracy improving. For Client-Client matching with distance threshold of 2, Exp1.1 detects 10,434 matches with an accuracy of 98.7%. However, with a similar accuracy of 99%, Exp3.3 loses 95 records (10,339). This appears to indicate a strong case for using contact details only.
- Overall, Exp2.2 was chosen as best because it included all the accurate matches and is efficient while constructing the similarity matrix. The number of true matches can be calculated by multiplying the number of matches (*Match*) by the accuracy percentage (*Acc %*). The total of true matches in Exp2.2 is 35848 ($Total \times Acc\%$). Across three matching rules: 10559 ($CC3 \times Acc\%$) accurate matches identified by the *Client-Client_Rule (C-C)*; there are 23220 ($CF2 \times Acc\%$) true matches identified from *Client-Family_Rule (C-F)* and 2069 ($CD0 \times Acc\%$) from *Client-Domicile_Rule (C-D)*.

The result for the unified records is shown in Table 5. Columns 2-4 represent the 3 types of matches: the *C-C* match, *C-F* match and *C-D* match. *Y* indicates if there are one or more matches for that match type and *N* for no relationship in this type. *Records* shows the number of records for that combination. In brief, there are 8 combinations and we can highlight some findings from the data regarding all the combinations. *Combination 1* for clients who are single policy holders; *Combination 2* to *Combination 4* are clients who have multiple policies for themselves or one for themselves and one or more policies for families or co-habitants; *Combination 5* to *Combination 7* are the clients involved in two types of relationships; finally, *Combination 8* are clients who had all three types of relationship.

In total there are 162,929 unified client records for a validation dataset of 194,396. Additionally, 30% of clients satisfied at least one of the relationship types.

Table 4: Results of Experiments by Threshold

	Exp1.1		Exp2.1		Exp2.2		Exp3.1		Exp3.2		Exp3.3	
Rules	Match	Accuracy	Match	Acc %	Match	Acc %	Match	Acc %	Match	Acc %	Match	Acc %
CC0	9609	99.95	9609	99.95	9609	99.95	9609	99.95	9609	99.95	9609	99.95
CC1	10146	99.70	10144	99.72	10144	99.72	10116	99.73	10113	99.73	10109	99.73
CC2	10434	98.73	10422	98.84	10418	98.88	10373	98.92	10354	99.04	10339	99.10
CC3	14649	72.08	13688	77.14	13493	78.26	12379	84.86	12054	87.09	11776	89.06
CF0	30330	72.14	30330	72.14	30330	72.14	30330	72.14	30330	72.14	30330	72.14
CF1	36057	64.12	35877	64.44	35856	64.48	32924	68.59	32692	68.99	32533	69.24
CF2	58754	39.52	57330	40.50	56695	40.96	40311	56.24	38775	58.39	37656	60.04
CD0	13270	15.59	13270	15.59	13270	15.59	13270	15.59	13270	15.59	13270	15.59
Total	86673	41.36	84288	42.53	83458	42.95	65960	53.43	64099	54.93	62702	56.08

Table 5: Unified Client Records

Combination	C-C	C-F	C-D	Records
1	N	N	N	137,114
2	Y	N	N	6,780
3	N	Y	N	14,174
4	N	N	Y	1,383
5	Y	Y	N	2,936
6	Y	N	Y	335
7	N	Y	Y	148
8	Y	Y	Y	59

6.2 Analysis

From Table 4, Exps 1.1, 2.1 and 2.2 performed best in terms of detecting most matches. The total figure, calculating by adding the best performing threshold experiments (CC3, CF2 and CD0) ranges between 83,458 and 86,673 although accuracy drops when detecting high numbers of matches. Of these, Exp2.2 is the most efficient due to the far lower number of comparisons required (see Table 3). This is to be expected as the blocking length increases and number of attributes reduced. Note that the overall accuracy is affected by the low accuracy for co-habitants (discussed later).

It is useful to note the numbers of dimensions used for matching (as opposed to segmenting) when discussing these results. In Client-Client matching, 4 dimensions are used; in Client-Family matching, 2 are used and for matching co-habitants only 1 dimension is used. Thus, the quality of matching will inevitably decrease as we discuss the different types of matches.

Our related research highlights the many approaches to record linkage and it is no surprise that, using a combination of these techniques, the *Client-Client_Rule* performance has the best accuracy across matches. The 0.05% (5) false matches that occurred in CC0, were as a result of the poor data quality for the *address* attribute. When providing address information, only 49% of clients provided the address detailed to door number and thus, all clients on the same street would be matched. The same quality issue for *Address* will result in false hits across all types of matches even where the distance threshold is set to 0.

If we look at experiments using all 9 attributes (and dimensional matrices) and the distance threshold is set to 1 (CC1) for all, then matches for all experiments increased by around 500 with a slight drop in accuracy to 99.7%. In effect, this means an extra 23 false matches Exp2.2 and these were found to be as a result of the client misspelling data somewhere in the 9 attributes. Again looking at Exp2.2, the reason 512 more records were matched between CC0 and CC1 is mainly due to misspelling names. In general, for Client-Client matches, there is a dramatic drop in accuracy when the distance threshold is increased from 2 to 3. In summary, based on both experimental efficiency, matches and accuracy, experiment 2.2 with threshold set to 2 (CC2) is the best configuration for *Client-Client* matching.

The *Client-Family_Rule* is generally not part of record linkage research. As expected, in sparse datasets (datasets with low numbers of client-client matches), the system detected more Client-Family matches. Interestingly, the optimum distance threshold is different. While we still have a significant change with threshold setting 2 and 3, there is enough deterioration in results between 1 and 2 to select a threshold setting of 1 (CF1). However, in Exp1.1, there were 30,330 matches detected with an accuracy of 72.14%. By increasing the distance threshold to 1, while this detects an extra 5,727 records, only 1,239 were accurate resulting in a drop in overall accuracy to 64.12%. For this category, it is not definitive if CF0 (all experiments are the same so choose 3.3 as the most efficient) or CF1 with more matches, but more checking and false positives (choose 2.2 as the most efficient combined with the higher matches).

The *Client-Domicile_Rule* did not perform well either on accuracy nor on the number of true matches.

The accuracy for co-habitants is very low even though the threshold was set to 0. The poor quality of *Address* is problematic for this match type, because *SM_Address* is the only similarity matrix used in this rule. Our fuzzy matching (threshold greater than 0) can handle abbreviations like 'rd' for 'road', 'st' for 'saint' in the *Client-Client_Rule* and *Client-Family_Rule* only because those rules required a higher dimensionality (used additional similarity metrics). In summary, while the number of false hits is high, it succeeded in providing a new dimension to the relationship graph for our industry partner.

7 CONCLUSIONS

Strategic business knowledge such as Customer Lifetime Values for a customer database cannot be delivered without building full customer records, which contain the entire history of transactions. In our work, we use real world customer datasets from the insurance sector with the goal of uniting client records by: connecting all records (various policy data) for the same client; connecting clients to family members (where both have policies); and connecting clients with co-habitants (where the co-habitant is also a client). As data is never clean, this is a significant task, even for relatively large datasets.

In this research, our goal was to segment the overall dataset so as to reduce matching complexity but to do so in a manner that kept "matching" records in the same segment. Early experiments were quite clear that an aggregated similarity matrix did not provide the required matching granularity to deliver accurate results. For this reason, we create a multidimensional similarity matrix and applied a set of rules to assist the matching process. Our results show very good matching results when comparing client-to-client data; quite good results when matching clients with family members and mixed results when trying to detect cohabiting policy holders. Evaluation was provided by our industry partner who, as a result of our work, are building far larger customer graphs (customer profiles) than was previously possible. For future work, our goal is to develop an auto-validation method to validate results and replace the current human checking process performed by our industry partners.

REFERENCES

Baxter, R., Christen, P., Churches, T., et al. (2003). A comparison of fast blocking methods for record linkage. In *ACM SIGKDD*, volume 3, pages 25–27. Citeseer.

Bhattacharya, I. and Getoor, L. (2007). Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):5.

Bilenko, M., Kamath, B., and Mooney, R. J. (2006). Adaptive blocking: Learning to scale up record linkage. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 87–96. IEEE.

Bilenko, M. and Mooney, R. J. (2003). Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 39–48. ACM.

Chen, I. J. and Popovich, K. (2003). Understanding customer relationship management (crm): People, process and technology. *Business Process Management Journal*, 9(5):672–688.

Cohen, W., Ravikumar, P., and Fienberg, S. (2003). A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation*, volume 3, pages 73–78.

Day, W. H. and Edelsbrunner, H. (1984). Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1):7–24.

Di Benedetto, C. A. and Kim, K. H. (2016). Customer equity and value management of global brands: Bridging theory and practice from financial and marketing perspectives: Introduction to a journal of business research special section. *Journal of Business Research*, 69(9):3721–3724.

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., and Zhang, W. (2014). Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM.

Etienne, B., Cheatham, M., and Grzebala, P. (2016). An analysis of blocking methods for private record linkage. In *2016 AAAI Fall Symposium Series*.

Ferguson, J., Hannigan, A., and Stack, A. (2018). A new computationally efficient algorithm for record linkage with field dependency and missing data imputation. *International journal of medical informatics*, 109:70–75.

Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

Hotho, A., Staab, S., and Stumme, G. (2003). Ontologies improve text document clustering. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 541–544. IEEE.

Huang, Z. (1998). Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data mining and knowledge discovery*, 2(3):283–304.

Kruskal, J. B. (1983). An overview of sequence comparison: Time warps, string edits, and macromolecules. *SIAM review*, 25(2):201–237.

Larsen, B. and Aone, C. (1999). Fast and effective text mining using linear-time document clustering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22. ACM.

Mamun, A.-A., Aseltine, R., and Rajasekaran, S. (2016). Efficient record linkage algorithms using complete linkage clustering. *PloS one*, 11(4):e0154446.

McCallum, A., Nigam, K., and Ungar, L. H. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178. ACM.

Pyle, D. (1999). *Data preparation for data mining*, volume 1. morgan kaufmann.

Rahm, E. (2016). The case for holistic data integration. In *East European Conference on Advances in Databases and Information Systems*, pages 11–27. Springer.

Sedding, J. and Kazakov, D. (2004). Wordnet-based text document clustering. In *proceedings of the 3rd workshop on robust methods in analysis of natural language data*, pages 104–113. Association for Computational Linguistics.

Yujian, L. and Bo, L. (2007). A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1091–1095.