Evaluating a system for facilitating the integration of heterogeneous data from different sources during crisis management

Tze Shuen Ng

4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2023

Abstract

The CHAIn (Combining Heterogeneous Agencies' Information) system facilitates information sharing between heterogeneous data sources by using matching to dynamically rewrite queries such that it succeeds on the target data source. Its unique approach means that comparisons to existing benchmarks are insufficient to constitute a thorough evaluation. This project uses crptr, a data corruption application, to approximate a gold standard matching to compare against matching produced by CHAIn to evaluate the correctness, selectiveness and completeness of its results.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Tze Shuen Ng*)

Acknowledgements

To Fiona McNeill, for her guidance and support throughout this project. My fortnightly meetings with her are the only reason I put pen to paper.

To my family, because I wouldn't be here without them, and for always providing a place to hide away.

To Brian, Mattias, and Junji for always lending a listening ear.

To Jakub, without whom I would have spontaneously combusted at some point this year, but especially for his friendship for the last 4 years.

Table of Contents

Chapter 1

Introduction

During crisis management, organisations share their data to get an overview of what is known and make critical decisions under immense time pressure, making fast and effective data sharing vital. However, successful querying of a data set requires a good understanding of that data set and its underlying schema to ensure that the query is correctly formatted. If data querying is part of an automated process, then prior knowledge of which data sources will be useful and the schema and data representation of each data set is necessary. Effective and automated data sharing is best achieved via pre-alignment of data sets from different sources; in an ideal situation, these data sets would also use a shared vocabulary for easy integration.

However, some of the organisations sharing data may be unknown, others may not wish to reveal their entire database schema for competitive reasons. Furthermore, data sources may be highly dynamic. Therefore, relying on pre-alignments during a crisis is not feasible: we cannot assume we will know exactly which organisations we will need to interact with in advance, what the context of the interaction will be, and what the schema and data representations of data sources will be at the time of querying. Furthermore, humans cannot efficiently deal with large data sources, hence it is not feasible manually identify mismatches in the schema and update queries accordingly when time is of the essence in an emergency.

Ideally, we would have a fully-automated pipeline which can operate without knowledge of the representation of the data by using a schema-agnostic query mechanism. Schema-agnostic or vocabulary-independent queries can be defined as query approaches over structured databases which satisfy complex information needs without a prior understanding of the representation of a structured database [3]. The application of a schema-agnostic query mechanism extends beyond emergency response scenarios, it can be used to mitigate heterogeneity across databases and expedite information sharing [4]. The CHAIn (Combining Heterogeneous Agencies' Information) system can be used by an owner of a data source to respond with the relevant information to incoming queries, even when these queries fail at the schema level and/or the data level[1] [11].

[1] A mismatch at the schema level occurs when the structure of the data differs, e.g. the column names are incorrect. A mismatch at the data level occurs when the query schema matches the data source schema, but the specific data differ, e.g. dates are given as DDMMYY rather than MMDDYY

To get a clear idea of how well the system is performing, we would ideally compare the results produced by CHAIn against some kind of gold standard matching for two data sources that are similar but mismatched in relevant ways. This gold standard should accurately and completely determine the correct mappings between these data sources and thus could be compared against mappings produced by CHAIn to objectively evaluate the quality of matches. Unfortunately, gold standards for dynamic query writing based on matching simply do not exist. Previous work on evaluation presented results from CHAIn to domain experts for review and asked them to assess if the results returned were relevant and helpful. Other approaches focused on adapting ontology matching benchmarks for evaluation [11]. However, neither approach is ideal. CHAIn was designed for a very different task, hence adapting ontology matching benchmarks which may not be entirely relevant to CHAIn is likely to be an incomplete evaluation. Creating our own gold standard mapping for CHAIn or getting experts to review results is labour-intensive and expensive. Even if we could develop a gold standard, there is always some level of subjectivity in mappings as experts sometimes disagree on the best matches.

1.1 Project Objectives

Since there are no good benchmarks to evaluate CHAIn, this project aims to *develop an approximation of a gold standard matching that will allow us to effectively evaluate the performance of CHAIn*. We then *compare the matches returned by CHAIn against this gold standard* to evaluate if the results returned are:

- Correct — if the matching also exist in the gold standard
- Complete — if all attributes in the incoming query were answered

The project will run CHAIn on data sets across various domains, using different syntax, to discover what areas CHAIn performs well, which areas could be improved upon, or if the system is not suitable for further development. Having the correct information is clearly necessary for decision-makers during a crisis, but the system should also be selective and only return the most relevant results. This enables the human user to quickly hone in on the most important results, making the task of sifting through data tractable. It is important to establish CHAIn's usefulness before progressing to the next stage of development, as there is little sense in extending, optimising, or beautifying a system that fails to return correct results.

Since there are no good benchmarks to evaluate CHAIn, we use the crptr system [10], a data corruption application which simulates data quality issues, to approximate a gold standard matching. We use it to modify a data set and create a clear mapping between the original and modified data set, which we use as a gold standard mapping to evaluate results from CHAIn.

The work done over the course of this project include:

- Set up CHAIn and crptr on a local machine.
- Develop a taxonomy of data sets and queries looking at different matching

challenges.

- Find data sets that fit our requirements.
- Create an approximate gold standard matching for each data set.
- Run queries on CHAIn to get results
- Analyse the results and evaluate CHAIn's performance on various types of mismatches

1.2 Overview

The structure of this dissertation is as follows:

- Chapter 2 provides the background on work done to develop the CHAIn and crptr systems, as well as some background into evaluating matching.
- Chapter 3 outlines the methodology used in this project: the process of creating a gold standard matching, the types of mismatches which we will use to query CHAIn, the evaluation metrics which we use to assess the results returned, and the data sets used in experiments.
- Chapter 4 details the results of the experiments, analysis of various mismatches where CHAIn performs well and pointing out its pitfalls
- Chapter 5 discusses the limitations of this project, outlines future extensions that could be done to evaluate other aspects of CHAIn, and provides a brief evaluation of crptr.
- Chapter 6 review any contributions made and concludes the paper.

Chapter 2

Background

In Section 2.1, we explain how CHAIn uses matching to determine whether there is anything in the schema of the data source that approximates to the schema of the query and responds to queries that may not be an exact match for the schema but for which it has relevant information. Section 2.2 describes the crptr system used to create our own mapping, which approximates a gold standard mapping, for evaluation purposes. Section 2.3 explains why crptr is necessary for this project: existing benchmarks in related fields such as ontology matching are not a good fit due to CHAIn's unique approach of dynamically rewriting queries such that they succeed on the target organisation's data sources.

2.1 CHAIn

Formulating correct queries during crisis management is extremely difficult. Queries from the querying organisation requesting data from the target organisation, which has data that may be relevant to the query, are likely to fail due to mismatches in the schema. CHAIn facilitates fast and effective information sharing during crisis management by enabling automated query reformulation. The system aims to broaden the range of queries to which target organisations can successfully return an appropriate response. CHAIn can respond not only to queries which are formatted with the correct terminology and structure for the target organisation's schema, but it can also respond to queries that may not be an exact match for the schema but for which it has relevant information. While it is in no way a replacement for proper planning and pre-alignment of data sources, it does allow for quick access to data sources that may not have been previously considered, or where data has been updated, leading to pre-alignments to fail.

Data and queries can vary in format, terminology, structure, specificity and organisation. Different organisations may store their data in RDF, relational database tables, spreadsheets, XML or text. CHAIn operates at the levels of structured and semi-structured data, such as relational databases and RDF, where the overall schema of the data sources and queries can be represented as first-order terms. The current implementation of CHAIn focuses on SPARQL queries to RDF data sources.

2.1.1 Motivating example

Consider an epidemic, where a health agency is monitoring the number of ill patients in the area. They have their data from NHS health boards, but would also like patient numbers from private clinics and hospitals that may be agreeable to sharing data during the crisis. Perhaps the agency's data sources has schema:

patients(healthboardID, count, date)

They will send this query to other organisations which they believe may have relevant information. If there is no mismatch at either the data or the schema level, the query succeeds and appropriate responses will be returned without the need to invoke CHAIn. However, the queried organisation may organise their data differently:

patientVisits(Date, total)

There are mismatches in the words used, the order of the arguments, and the number of arguments. The query from the health agency will fail because it is not correctly formatted with respect to the data source schema.

2.1.2 Query response process

If a query succeeds, there is no need for CHAIn to intervene. When a query fails, CHAIn will first check if the query is correctly formatted per the schema of the target data source. If not, the schema of the query is matched against the target data source schema to find possible alignments. CHAIn rewrites the query so that it is semantically similar or even equivalent to the original query, but is now also aligned with the target data source schema. The matching part of the process is done by the Structure-Preserving Semantic Matching algorithm (SPSM) [7] which does pairwise matching on structured terms. In most cases, there will be several approximate matches instead of one identical match for the incoming query. CHAIn will assign all matches a similarity score and all matches that cross a certain threshold are ranked according to the similarity score and then returned to the querying organisation (see Figure 2.1). A human user will able to quickly hone in on the best response if CHAIn successfully returns a small number of highly relevant responses ranked accordingly.

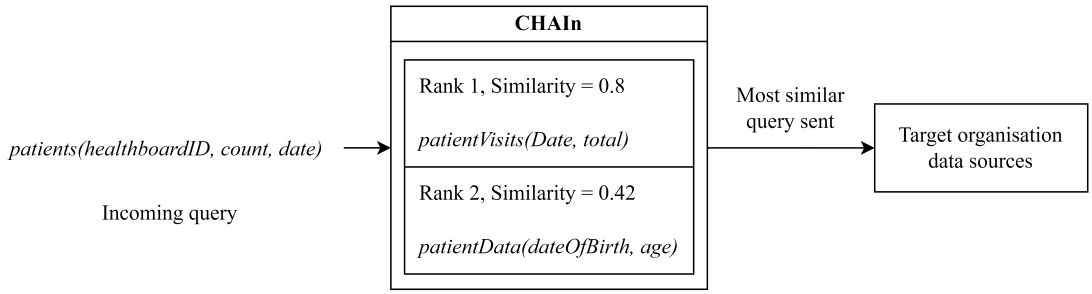

Figure 2.1: Example of rewritten queries by CHAIn

Life cycle of the process:

1. A query is received by the target data source.

2. If the query fails, CHAIn determines if this is due to some mismatch at the schema level. If so, naive matching is done on the schema of the target data source to narrow down the search space to likely matching. If not, skip ahead to step 5.
3. Potential matches are produced by the SPSM process. Each match is ranked according some assigned score.
4. The query is rewritten according to the matches and sent to the target data source.
5. If the query fails again, we look for mismatches at the data level since the query is correctly formatted in accordance with the target data source's schema at this point.
6. Potential responses to the query that meet a certain threshold are returned, along with some description of how CHAIn produced the matching.
7. A human user with knowledge of the data source can choose any number of matches to be returned to the querying organisation. If no human is involved, results can be automatically sent to the querying organisation based on their ranking.
8. The querying organisation receives the response(s) to their original query, together with information about matches that were used to produce the response.

2.1.3 Narrowing down data sources

It is not feasible to use the SPSM algorithm to conduct pairwise matching between a single query and every possible data set owned by the receiving organisation (the organisation might have a large number of data sources) since it is an expensive process. Therefore, in step 2 of the query response process, CHAIn narrows down the search space by performing keyword matching on the subject of the query to the data sources owned by the receiving organisation. Filtering is done with two generic ontologies/lexographies: SUMO and WordNet [12]. In more detail, step 2 of the query response process is as follows:

2.1 After a query fails, we compute Query Words set, which is the set of words that appear in the query schema.

2.2 Compute the Related Terms set, which is the set of words that are semantically similar to words in the Query Words set, based on the SUMO and Wordnet ontologies.

2.3 Compute the Target Words set, which is the set of words that appear in data set names owned by the receiving organisation.

2.4 Find overlapping words that exist both in the Related Terms set and the Target Words set to identify relevant data sources.

2.5 Search the target data source for any predicates with any of these names and return them.

Going back to the motivating example, when the health agency queries a private clinic, the narrowing down process would look like this:

Query words	patients
Related terms	. . . , sufferer, patient, patience, . . .
Target words	patient, visits
Overlapping words	patient
Data sets returned	patientVisits, patientData

2.1.4 Finding matches

CHAIn operates on data sources and queries which can be represented as first-order terms [11]. Step 3 of the query response process finds potential matches between the query and data source schema using the SPSM algorithm. CHAIn constructs two trees from the first-order term representing the incoming query and the first-order term representing the target data source and passes these two trees to the SPSM algorithm. The SPSM algorithm uses the S-Match ontology matching system [8] to investigate relationships between individual words and keywords in the query schema and the data source schema. It also matches the structure of the trees to try to discover an overall relationship, which is important when considering if two terms are similar [7]. The SPSM algorithm is able to demonstrate whether the two trees are globally similar, and returns the mappings of each node of the source tree to nodes in the target tree or to null, along with explanations of how similar they are and in what way. A similarity score in [0 1] is returned, with 1 indicating that the query and data source schema are identical.

2.2 crptr

crptr is a data corruption application developed to simulate data quality issues for test data sets used for record linkage evaluation. It mimics the corruptions (errors and variations) commonly found in data sets and enables the user to control data quality by simulating and injecting data corruption into data sets using non-random methods that simulate the real-world errors and variations that occur in data [1]. To evaluate CHAIn, we focus on corrupting data sources and generating corrupted queries from those corrupted data sources. Previous work has already adapted crptr for CHAIn which removed irrelevant corruption methods and adapted some methods to better suit our use case [10]. Further mentions of crptr in this project shall refer to the modified version of crptr.

crptr modifies the data set in ways that are a plausible representation of naturally occurring mismatches in the schema of data sets. It can structurally corrupt the data by removing or adding attributes to the schema and by reordering attributes. To simulate semantic heterogeneity, it is also able to use lexical resources to determine semantically related terminology that could plausibly be used in similar domains to rename attributes, or even split or merge attributes. The current implementation uses WordNet [12], as it is by far the biggest and most widely used general lexical resource, although it would also be possible to plug in domain-specific lexicons and ontologies to facilitate the evaluation of domain-specific matchers.

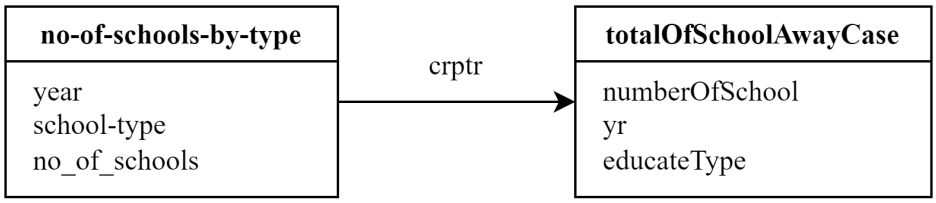

Figure 2.2: Example of a corrupted schema

2.3 Evaluating matches

One common method of evaluating matching systems is benchmarking. A benchmark is a well-defined set of tests on which the results of a system can be measured [5]. In the ideal scenario, we would have a gold standard matching for two data sources that are similar but mismatched in relevant ways. This gold standard should accurately and completely determine the correct mappings between these data sources, and thus could be compared against the results of the matching system to objectively evaluate the quality of matches.

2.3.1 Ontology matching

An ontology typically provides a vocabulary describing a domain of interest and a specification of the meaning of terms in that vocabulary. Ontology matching aims at finding correspondences between semantically related entities of different ontologies [2].

Disparities between data sources arise because organisations are disconnected and autonomous entities, each building its own IT infrastructure [9]. Individual designers also have different perspectives and may choose to represent data differently. This leads to different naming conventions, semantic interpretations of data and constraints applied [6]. This leads to heterogeneity between different data sources even though the information stored is related conceptually. The goal of matching ontologies is to reduce heterogeneity between them or make it possible to translate between them.

CHAIn is related to the field of ontology matching because the purpose of CHAIn is to map the semantics of a query into the target data sets, i.e. match the schema of the query to the schema of the target data source. Previous work on evaluating CHAIn relied on the well-researched field of ontology matching for benchmarks to evaluate CHAIn on.

2.3.2 Ontology Alignment Evaluation Initiative (OAEI)

The Ontology Alignment Evaluation Initiative (OAEI)[1] organises evaluation campaigns aimed at evaluating ontology matching technologies. Previous efforts at evaluating CHAIn used the Conference Set – one of the evaluation sets that are used – which contains data sets that are all about a similar topic, but each data set varies in terminology and structure. Each data set ontology was used to form possible data source schemas or queries to be matched, which were then used to evaluate matches between two data sets.

[1] http://oaei.ontologymatching.org

However, the OAEI was intended for evaluating ontology matching, which deals with fully matching one ontology to another. This is slightly different from CHAIn, which is usually only concerned with a subset of a much larger data source when rewriting failed queries using matching. Thus, while it is possible to adapt the OAEI, as the only high-quality expert-led matching gold standard, to evaluate CHAIn, we must bear in mind that the OAEI was not designed for this task. Such an adaptation will be limited and create only a partial evaluation of the CHAIn system.

Furthermore, CHAIn is interested in domain-aware matching, but developing these benchmarks is laborious, time-consuming, and requires experts in the domain of the data source who also have a good understanding of knowledge representation [15]. Even after meticulously developing a gold standard, there is always some level of subjectivity in mappings as experts sometimes disagree on the best matches. There simply are not many benchmarks we can adapt, and they cannot cover a large range of domains. Due to CHAIn's unique approach of using failure-driven reasoning techniques combined with matching techniques to dynamically rewrite queries, we require an alternative solution for our evaluation.

Chapter 3

Methodology

The purpose of this work is to evaluate the quality of the rewritten queries produced by CHAIn across multiple domains. Section 3.1 outlines to process of converting .csv data sets to RDF graphs since .csv data sets are far more available. Section 3.2 lays out how crptr creates a corrupted data set and a mapping back to the original data set. This mapping is compared against the mapping of the original query to the rewritten query produced by CHAIn during evaluations. We outline the types of corruptions introduced by crptr to evaluate CHAIn's ability to correct structural and semantic mismatches in Section 3.3. The order in which queries are run is described in Section 3.4. The evaluation metrics and data sets used in this project are introduced in Section 3.5 and 3.6 respectively.

3.1 Creating RDF data sources

CHAIn currently works on RDF data sources. However, data sets in RDF format are less available, so we use Tarql, a command-line tool which converts the data to semantic triples ready to load into a knowledge graph to convert using SPARQL 1.1[1] to create a process of converting .csv data sets to .ttl RDF graphs.

Life cycle of the process

1. Clean up the headings of the data set, e.g. remove white space and illegal characters in SPARQL like "." .
2. Define a mapping to convert the .csv file into RDF triples. The mapping indicates how each column will be mapped with a predicate, which file are we trying to convert, and how we can apply datatypes to some of the variables.
3. Call Tarql to construct a RDF graph from the .csv data set.

[1] http://tarql.github.io/

3.2 Producing a gold standard

crptr takes the data set and corrupts it structurally and semantically at the schema level to produce a new schema where each attribute can be mapped to some attribute in the original data set (Figure 2.2). This creates two mismatched data sources that are about a similar topic and a clear mapping back to the original data set, which we use as our gold standard matching to evaluate matches produced by CHAIn.

We use the number of columns in a data set and the number of modifications made by crptr as indicators of the difficulty of mapping the corrupted schema to the original schema — more columns and more modifications indicate a harder task for CHAIn. Since the data sets used in this project are fairly small, we evaluate at two levels of difficulty: once at full difficulty with every word modified, and again with half of the words present in the schema modified. For each difficulty level, we use crptr to create three corrupted schemas and record CHAIn's performance against each schema.

Life cycle of the process:

1. Each of the headings are split up into individual words. crptr creates a lookup table from the split up words from the headings of the data set which are sent to WordNet and then returned as a list of synonyms. Using the example given in Figure 2.2, crptr creates the following lookup table:

Headings	Synonyms
school	schoolhouse
school	educate
school	train
no	total
no	number
year	yr
year	class
type	case
type	typewrite
by	past
by	away
⋮	⋮

2. At full difficulty, every word in the data source schema is modified by some corruption function. For the easier data set, half of words are selected at random for corruption. The corruption functions available in crptr are:

   (a) synonyms_corruptor uses the created lookup table of synonyms to make the modifications to each of the word in heading names.

   (b) ocr_corruptor uses a list of similar pairs of characters or strings that will be applied on words in the heading names, e.g. '0' is modified to 'o'.

   (c) name_abbr shortens the word to a specified length of characters, e.g. 'School' could be shortened to just 'S'.

The probability of the corruption method used to modify some word in the schema is as follows:

Corruption method	Probability
synonyms_corruptor	0.4
ocr_corruptor	0.3
name_abbr	0.3
Total	1

The headings are corrupted at random and a corrupted schema and list of queries for that corrupted schema is generated. Using the corrupted schema produced in Figure 2.2, an example of a corrupted query is

totalofSchoolAwayCase(numberOfSchool, yr, educateType)

3.3 Types of mismatches

Corrupted queries should encompass a variety of schematic differences so that we evaluate CHAIn thoroughly. Here we outline the types of mismatches used to evaluate CHAIn.

3.3.1 Semantic differences

[6] outlines some descriptions of measures of semantic relationships, which we shall adopt to ensure that the corrupted queries include a variety of semantic differences:

1. Identical — where there is no evidence of incoherence, i.e. our query schema is not corrupted and matches the data source schema. Such mappings should be identified by CHAIn.
2. Equivalent — where representations are not identical but concepts are the same and coherent, e.g. schemaA.Airlines versus schemaB.EuropeanAirlines, both model the same information, but the location is not implied in schema1. The performance of CHAIn depends on its ability to correctly find semantic mappings between the equivalent representations.
3. Incompatible — where the data representation is incoherent. CHAIn should realise that the attribute does not exist in the data set and return no results.

It is a failure for CHAIn to return responses when all attributes in a query are incompatible with the data present in the target organisations data sources. Likewise, it is a failure to return results if there are equivalent mappings to headings in the data source.

The similarity score computed in step 3 of CHAIn's query response process is used to determine if a query is identical, equivalent, or incompatible with the data source schema. A similarity score of 1.0 suggests that the query schema is identical to the data source schema, i.e. that every single attribute in the query matches exactly to some column in the data source. Queries with similarity scores below a threshold of 0.2 are classified as incompatible, and CHAIn will return no results. We set the threshold to be fairly low because we are also interested in analysing the matches produced by CHAIn

and want to view the full range of results returned. Queries with scores between the threshold and 1.0 are classified as equivalent queries since there is some equivalent mapping between the query schema and the data source schema.

3.3.2 Structural differences

Structural differences occur when the structure of the query schema differs from the data source schema, e.g. the order of arguments in the query does not match order of the columns of the data set, the number of arguments in the query differs from the number of attributes in the data set, or the data set uses multiple columns or subcategories to represent the same information. We include corrupted queries where column are omitted, out of order, or split/combined in order to evaluate CHAIn's performance on structural corruptions.

3.4 Procedure

For each data set, we run queries in the following order:

1. An identical query where no attribute is modified.
2. Equivalent queries where every attribute is modified by crptr, but the order of attributes is maintained. This is allows us to focus on CHAIn's ability to correct semantic mismatches, i.e. if CHAIn fails to rewrite a query, it is due to some deficiency with correcting semantic mismatches. For example, using the corrupted schema example in Figure 2.2, for a data source with schema

   no-of-schools-by-type(year, school_type, no_of_schools)

   We run the corrupted query

   totalOfSchoolAwayCase(yr, educateType, numberOfSchool)

   which modifies every attribute in the data source but preserves the order.
3. Equivalent queries that feature semantic and structural mismatches. We use semantic corruptions that were evaluated under step 2 to control for semantic mismatches. If CHAIn was able to find a suitable mapping for some modified attribute in step 2, but fails in step 3, we will know that it is due to some problems with resolving structural mismatches. An example of a corrupted query run during this step is

   totalOfSchoolAwayCase(numberOfSchool, yr)

   which features the same semantic corruptions as the query in step 2, but has additional structural mismatches such as missing attributes, and attributes in the wrong order.
4. Lastly, we run queries that feature semantic mismatches, strutural mismatches, as well as incompatible mappings, i.e. attributes in the query that do not map to any heading in the data source, and should be mapped to null. An example of such a query would be

totalOfSchoolAwayCase(educateType, zipcode)

3.5 Evaluation measures

The commonly accepted measures for qualitative matching evaluation are based on well-known information retrieval measures of relevance, such as Precision and Recall [17]. The calculation of these measures is based on the comparison between the matches produced by a CHAIn (denoted C) and our gold standard matches produced by crptr (denoted G). Finally, we denote the set of all possible matches, namely the cross product of the attributes of the two data sets, as A (Figure 3.1).

Correct matches by CHAIn are True Positives (TP) and can be computed as follows:

$$TP = C \cap G$$

Incorrect matches by CHAIn are False Positives (TP) and are computed as follows:

$$FP = C - C \cap G$$

Correct matches missed by CHAIn are False Negatives (FN) and are computed as follows:

$$FN = G - C \cap G$$

Incorrect matches not returned by CHAIn are True Negatives (TN) and are computed as follows:

$$TN = A - C \cup G$$

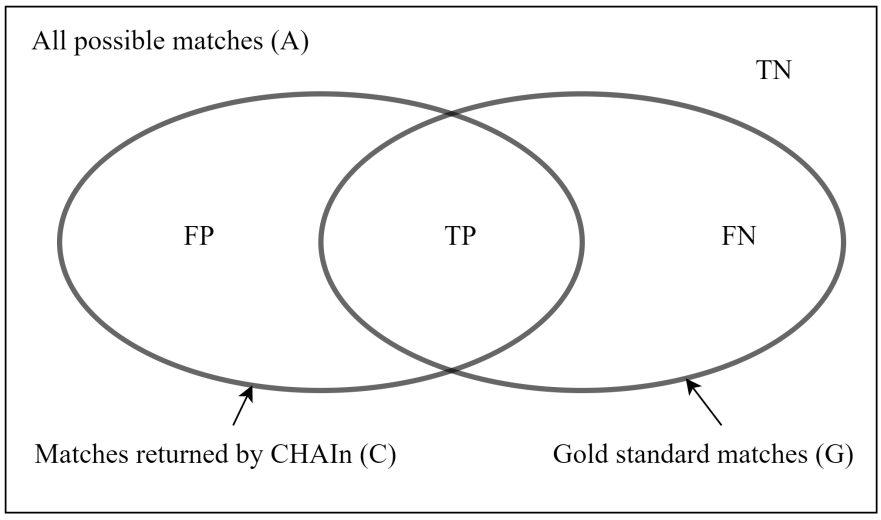

Figure 3.1: Basic set of matches

Precision is the ratio of matches correctly identified by CHAIn to all matches found by CHAIn. It is a measure of correctness: the higher the value, the smaller the set of false positives identified. It is calculated as follows:

$$Precision = \frac{TP}{TP + FP} = \frac{C \cap G}{C}$$

Recall, sometimes called Sensitivity, is the ratio of matches correctly identified by CHAIn to all matches in the gold standard. It is a measure of completeness: the higher the value, the smaller the set of true positives which have not been found. It is calculated as follows:

$$Recall = \frac{TP}{TP+FN} = \frac{C \cap G}{G}$$

It is insufficient to evaluate CHAIn solely using Precision or Recall. It is trivial to achieve perfect Recall simply by returning all possible matches, although this causes Precision to be very low since the set of matches by CHAIn, C, will be very large. Similarly, Precision can be maximised at the expense of Recall by only returning a few correct matches. We need to consider both measures simultaneously using a combined measure such as the F-measure.

The F-measure is the harmonic mean of Precision and Recall, meaning that both measures will be evenly weighted. Hence, the F-measure is a global measure of the matching quality and allows us to compare results while considering both Precision and Recall. It is calculated as follows:

$$F - measure = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

3.6 Data sets

Matching tasks are extremely varied, a system can perform very well on some data and give terrible results on others [2]. Evaluation data sets should therefore be as varied as possible so we can assess CHAIn's performance on different tasks. To evaluate CHAIn's usefulness in the real world, we sourced publicly available data sets across various domains and corrupted them using crptr. We looked for data sets that were well-written i.e. used proper English to establish expected performance on most data sets. We also used data sets with various syntaxes, e.g. presence of punctuation and symbols, the use of acronyms etc. to evaluate the effects of different syntax on CHAIn's performance.

3.6.1 Trees_In_Camden

Headings: NumberOfTrees, Sequence, SiteName, ContractArea, ScientificName, CommonName, InspectionDate, InspectionDueDate, HeightInMetres, SpreadInMetres, DiameterInCentimetresAtBreastHeight, Maturity, PhysiologicalCondition, TreeSetToBeRemoved, RemovalReason, NewlyPlanted, OutstandingJobCount, OutstandingJobNumber, OutstandingJobDescription, CapitalAssetValueForAmenityTrees, CarbonStorageInKilograms, GrossCarbonSequestrationPerYearInKilograms, PollutionRemovalPerYearInGrams, WardCode, WardName, Easting, Northing, Longitude, Latitude, Location, Identifier, SpatialAccuracy, LastUploaded, OrganisationURI

The data set[2] contains data about trees scattered around Camden. It is the largest data set sourced with 34 headers and 66 unique words present in the schema. Attribute

[2] https://opendata.camden.gov.uk/resource/csqp-kdss.csv

names are written well, although some are rather long e.g. ”GrossCarbonSequestrationPerYearInKilograms”. There are a few words that are frequently repeated throughout the data set: “name” appears 4 times, ‘tree” / “trees” appears 3 times.

3.6.2 Cool_It_NYC_2020_-_Drinking_Fountains

Headings: System, DFActivated, FountainType, FountainCount, Painted, FilterInstalled, Position, Outdoor, GISPropNum, PropertyName, OMPPropID, SubPropertyName, ParkDistrict, Borough, AMPSID, AMPSClass, AMPSStatus, AMPSParentID, AMPSName, CommunityBoard, CouncilDistrict, Precinct, ZIPCode, x, y

The data set[3] is a list of drinking fountains around New York City. It has 25 columns and 40 unique words present in the schema. Headings are well-written, although there are general acronyms like “ID” and “Num” and also domain / organisation specific acronyms e.g. “AMPS” and “GIS”.

3.6.3 NASA_Facilities

Headings: Center, CenterSearchStatus, Facility, FacilityURL, Occupied, Status, URLLink, RecordDate, LastUpdate, Country, Contact, Phone, Location, City, State, Zipcode

The data set[4] lists NASA Facilities. There are 16 columns and 20 unqiue words present in the data set. The attribute names are short and well-written. The only acronym present is “URL”, which is fairly common and included in WordNet.

3.6.4 Apple-Twitter-Sentiment

Headings: _unit_id, _golden, _unit_state, _trusted_judgments, _last_judgment_at, sentiment, sentiment_confidence, date, id, query, sentiment_gold, text

The data set[5] looks at the sentiment around Apple, based on tweets on Twitter. There are 12 columns and 17 unique words present in the data source schema. Many headers use underscores at the start of the header or between words. There many similar headings e.g. ”sentiment”, ”sentiment_confidence”, ”sentiment_gold”.

3.6.5 gene2go

Headings: index, tax_id, GeneID, GO_ID, Evidence, Qualifier, GO_term, PubMed, Category

The data set[6] reports the GO terms that have been associated with Genes in the Entrez Gene database. There are 9 columns and 11 unique words present. This data set uses domain-specific terminology like gene, tax_id (short for taxonomy ID), GO term (the

[3] https://data.cityofnewyork.us/resource/wxhr-qbhz.csv
[4] https://data.nasa.gov/resource/gvk9-iz74.csv
[5] https://data.world/crowdflower/apple-twitter-sentiment
[6] https://ftp.ncbi.nlm.nih.gov/gene/DATA/gene2go.gz

term in the Gene Ontology), and PubMed (a database of biomedical literature). It would be interesting to see CHAIn's performance on a very domain specific data set.

3.6.6 graduates-from-university-first-degree-courses-by-type-of-course

Headings: year, sex, type_of_course, no_of_graduates

The data set[7] records the number of university graduates in each course. There are 4 headings and 9 unique words present in the data source schema. Words are separated by a dash in the headings, and there are no repeated words present.

3.6.7 average-daily-traffic-volume-entering-the-city

Headings: year, ave_daily_traffic_volume_entering_city

The data set [8] has 2 columns and will be used to in conjunction with the average-speed-during-peak-hours data set (Section 3.6.8) to evaluate CHAIn's ability to resolve structural mismatches due to data being stored in multiple data sources.

3.6.8 average-speed-during-peak-hours

Headings: year, ave_speed_expressway, ave_speed_arterial_roads

The data set [9] has 3 columns and will be used to in conjunction with the average-daily-traffic-volume-entering-the-city data set (Section 3.6.7) to evaluate CHAIn's ability to resolve structural mismatches due to data being stored in multiple data sources.

[7]https://data.gov.sg/dataset/graduates-from-university-first-degree-courses-by-type-of-course
[8]https://data.gov.sg/dataset/road-traffic-conditions
[9]https://data.gov.sg/dataset/road-traffic-conditions

Chapter 4

Results and Evaluation

Section 4.1 details CHAIn's performance when matching data set names. The evaluation metrics of equivalent queries with no structural mismatches are documented in Section 4.2. Sections 4.3 and 4.4 describes the various types of mismatches where CHAIn does well, and where it falls short. Section 4.5 briefly summarises the results and discusses CHAIn's performance overall.

4.1 Keyword matching is inflexible

After a query failure, the process of narrowing down the search space (described in Section 2.1.3) begins when a query fails due to a mismatch in an attribute, even if the subject of the query is an exact match for a data source in the target organisation. For example, if the following query is sent to the data set flowerInGarden.ttl:

flowerInGarden(NotAnAttribute)

It fails because there is no column called NotAnAttribute in the data set. This starts the narrow-down procedure to discover if there is a similar match among the target organisations' data sets.

Query words	flower, In, Garden
Related terms	. . . , flower, in, IN, garden, gardening, . . .
Target words	in, garden, flower
Overlapping words	in, garden, flower
Data sets returned	flowerInGarden

All target words appear in the set of related terms, so the narrow-down process is still able to find the correct data set and rewrite the query such that it succeeds on the data source. Generally, the narrow-down process works well because query words tend to appear as related terms anyway. However, there are examples when this process fails. For example, the following query fails when it is sent to the data source treesInCamden.ttl:

treesInCamden(NotAnAttribute)

The narrow-down process computes the following sets of words:

Query words	trees, In, Camden
Related terms	. . . , in, Tree, tree, Camden, . . .
Target words	in, camden, trees
Overlapping words	in
Data sets returned	There are no matches returned.

The word "Camden" appears in the related terms set with its first letter capitalised since it is a proper noun. However, by default, all words in the target words set are lowercase, hence keyword matching fails to recognise that these words overlap. The word "trees" does not count as an overlapping word because plural forms aren't included in the WordNet library, hence it is not included in related terms. Hence, the keyword matching algorithm is unable to match "tree" to its plural form "trees".

Keyword matching also fails when there is punctuation, symbols or digits present. For example, a query to Cool_It__NYC_2020_-_Drinking_Fountains.ttl has the following result:

Cool_It__NYC_2020_-_Drinking_Fountains(NotAnAttribute)

Query words	cool_, nyc_2020_-_, it__, drinking_, fountains
Related terms	. . . , coolheaded, alcoholic_drink, fountain, IT, . . .
Target words	cool_, nyc_2020_-_, it__, drinking_, fountains
Overlapping words	
Data sets returned	There are no matches returned.

We see that symbols and digits are not removed when computing the target words set. Since none of the related terms, which were taken from the WordNet library, have any symbols or numbers in them, keyword matching does not find any matches. The only target word that only contains the alphabet is "fountains". However it is a plural form, and as explained earlier, will not count as an overlapping word even though its root word "fountain" appears in the set of related terms. Hence we have an example of keyword matching failing to find any overlapping words even though the subject of the query is an exact match.

4.1.1 Improvements and mitigation measures

It is quite clear that using keyword matching to narrow down the search space is rather inflexible and does not do well with plural forms, proper nouns, and the presence of symbols and digits. These occur frequently in database schemas (all of the data sets used in this project ran into issues of some form), hence this is a serious shortcoming of CHAIn's procedure to narrow down the search space. However, it should not be too difficult to mitigate this issue. Future work could add plural forms to the set of related terms, convert all related terms to lowercase before finding overlapping words (so that Proper nouns do not start with capital letters), and remove symbols and digits when identifying words.

These improvements are outside the scope of this project. To get CHAIn to successfully

return results despite inflexible keyword matching, we simplify the data set names by removing all symbols and numbers, proper nouns, and plural forms so that CHAIn can find the target data set (Table 4.1). Data set names are also shortened to simplify the matching process described in Section 2.1.4. The rest of this project will focus on evaluating CHAIn's performance when finding mappings between attributes in the incoming query and columns in the data source. Subsequent queries shall not modify the data set name, only featuring structural mismatches and semantic mismatches in the attributes.

Original data set name	Simplified data set name
Trees_In_Camden	tree
Cool_It_NYC_2020_-_Drinking_Fountains	drinkingFountain
NASA_Facilities	facility
Apple-Twitter-Sentiment	appleTwitterSentiment
gene2go	geneToGo
graduates-from-university-first-degree-courses-by-type-of-course	universityDegree
average-daily-traffic-volume-entering-the-city	traffic
average-speed-during-peak-hours	speed

Table 4.1: Simplified table names to simplify keyword matching.

4.2 Results

We record the number of columns and the number of modifications[1] in a data set as some measure of the difficulty of the matching task in Table 4.2. We measure the Precision, Recall, F-measure and similarity score given by CHAIn for all queries run during step 2 of the experiment procedure, i.e. equivalent queries with no structural mismatches.

Results indicate moderate levels of Precision ranging between 0.5-0.8 on data sets that have the maximum number of modifications, but Recall is usually less than 0,5. This causes the F-measure to be fairly low as well, usually falling below 0.5. CHAIn always performs better on the data set with fewer modifications: Precision is always above 0.75 while Recall usually ranges between 0.5 to 0.8. The F-measure is also remarkably higher on the easier data sets, never falling below 0.6. This suggests that CHAIn can account for a few mismatches in the schema, but not if the query schema and data source schema is entirely different.

Subsequent sections will examine various types of mismatches and provide examples that are representative of CHAIn's performance when resolving each type of mismatch.

[1]Modifications are only made to the headings of each data set. The underlying data is not relevant to this project and thus not modified.

Data set	No. of columns	No. of modifications	Precision	Recall	F-measure	Similarity (from CHAIn)
tree	34	66	0.455	0.147	0.222	0.271
			0.727	0.235	0.356	0.3
			0.8	0.235	0.364	0.286
		33	0.96	0.706	0.814	0.671
			0.923	0.706	0.8	0.686
			0.95	0.559	0.704	0.543
drinkingFountain	25	40	0.5	0.12	0.194	0.231
			0.667	0.24	0.353	0.308
			0.375	0.12	0.182	0.308
		20	1	0.8	0.889	0.75
			0.882	0.6	0.714	0.596
			0.824	0.56	0.667	0.654
facility	16	20	0.625	0.313	0.417	0.441
			0.5	0.188	0.273	0.353
			0.667	0.5	0.571	0.706
		10	0.917	0.688	0.786	0.618
			0.714	0.625	0.667	0.824
			0.75	0.563	0.643	0.706
appleTwitterSentiment	12	17	0.8	0.667	0.727	0.692
			0.8	0.667	0.727	0.615
			0.625	0.417	0.5	0.462
		8	0.818	0.75	0.783	0.846
			0.8	0.667	0.727	0.731
			0.778	0.583	0.667	0.692
geneToGo	9	11	0.333	0.111	0.167	0.300
			1	0.333	0.5	0.35
			1	0.111	0.2	0.200
		5	1	0.556	0.714	0.6
			1	0.667	0.8	0.7
			1	0.444	0.615	0.5
universityDegree	4	9	1	0.25	0.4	0.3
			1	0.25	0.4	0.4
			1	0.25	0.4	0.4
		4	1	0.75	0.857	0.7
			1	0.5	0.667	0.5
			1	0.5	0.667	0.6

Table 4.2: Results of equivalent queries with no structural mismatches.

4.3 Semantic mismatches

4.3.1 Identical queries

Identical queries (where the schema of the incoming query is an exact match for the data source schema) are not an issue for CHAIn. For all data sets, identical queries were returned with F-measure = 1 and a similarity score of 1.

4.3.2 Identical matches

Identical matches are individual attributes that exist in both the incoming query and the data source. CHAIn sometimes fails to identify identical matches in corrupted queries. This usually occurs when there other headings in the data source that have similar words. For example, the query *tree(sName, plebeianName, wardName)* querying the *tree* data source produces mappings found in Table 4.3. The attribute *wardName* is not matched correctly even though it is an exact match for the heading *WardName*.

Original data source heading	Corrupted attribute in query	Match produced by CHAIn
WardName	wardName	CommonName
ScientificName	sName	SiteName
CommonName	plebeianName	ScientificName

Table 4.3: CHAIn failing to find identical matches

Instead, we see that CHAIn appears confused by a few headings that contain the word "name". This suggests that the SPSM algorithm is not selective enough to differentiate between attributes/headings that contain the same word.

4.3.3 Equivalent matches: different forms of the same root word

Unlike keyword matching used in the process of narrowing down the search space, the SPSM algorithm used during matching is more sophisticated. CHAIn has little issue matching query attributes to data source headings when they are different forms of the same root word.

We denote correct matches from query attributes to data source returned by CHAIn as sequences $\xrightarrow{CHAIn}$ Sequence, i.e. CHAIn correctly matches query attribute *sequences* to the data source heading *Sequence*. Examples below shows CHAIn correctly matching words in different forms:

- lUpload $\xrightarrow{CHAIn}$ LastUploaded
- newPlant $\xrightarrow{CHAIn}$ NewlyPlanted
- sequences $\xrightarrow{CHAIn}$ Sequence

4.3.4 Equivalent matches: synonyms

CHAIn performs well at finding synonyms. For most examples where the query attribute and data source heading are one-word synonyms, CHAIn can correctly identify matches. An example of this type of mismatch is:

- class $\xrightarrow{CHAIn}$ Category.

Headings that contain multiple words have more modifications introduced by crptr, and so we expect these matches to be harder for CHAIn. While true that there are fewer examples of CHAIn successfully matching longer attributes and queries, there are still a few examples which indicate potential :

- urban_center $\xrightarrow{CHAIn}$ City
- reviewAppointment $\xrightarrow{CHAIn}$ InspectionDate

4.3.5 Equivalent matches: acronyms

Acronyms in the test data sets include commonly used acronyms like “id”. Since WordNet is used to extract semantic labels for each word, we expect CHAIn’s performance on common acronyms to be similar to that of synonyms if these acronyms also exist in WordNet. Unfortunately, this does not appear to be the case. Results suggest that CHAIn struggles at matching commonly used acronyms:

- CarbonStorageInKg $\overset{CHAIn}{\not\longrightarrow}$ CarbonStorageInKilograms
- facilityUniform_resource_locator $\overset{CHAIn}{\not\longrightarrow}$ FacilityURL

CHAIn was unable to match *CarbonStorageInKg* to *CarbonStorageInKilograms* nor *facilityUniform_resource_locator* to *FacilityURL*. This suggests that WordNet either does not include acronyms, or it does not contain enough semantic information about acronyms for the SPSM algorithm to find the correct match.

For organisation/domain-specific acronyms, we expect CHAIn to give poor results since WordNet will not contain semantic information about the term and would struggle to find a match. It is therefore a pleasant surprise that we have result GeneOnotology_term $\xrightarrow{CHAIn}$ GO_term is correctly matched since GO is a domain-specific acronym.

4.3.6 Equivalent matches: abbreviations

The corruptions are introduced by the name_abbr corruption function are used to simulate abbreviations found in schemas. CHAIn fails every time the entire query attribute is a single-letter abbreviation of the data source heading. An example of this type of mismatch is:

- c $\overset{CHAIn}{\not\longrightarrow}$ Country

This is probably because the SPSM algorithm does not have semantic information about these abbreviations since they do not appear in WordNet. Hence, it is unable to identify similarity between the query attribute and data source headings and produce a matching.

However, CHAIn is usually able to find matches for headings with abbreviations if they contain more than one word:

- rDate_stamp $\xrightarrow{CHAIn}$ RecordDate
- sName $\overset{CHAIn}{\not\longrightarrow}$ ScientificName

The first example succeeds probably because CHAIn is good at matching synonyms and successfully mapped "Date_stamp" to "Date".

Another thing to take note about matching abbreviations is that they are difficult even for a human. In the second example, sName $\xrightarrow{CHAIn}$ SiteName, CHAIn matched *sName* to *SiteName* instead of the correct heading, *ScientificName*. This error probably should be tolerated since without access to the mapping produced by crptr, both *ScientificName* and *SiteName* would be reasonable matches.

4.3.7 Equivalent matches: misspellings

CHAIn does not deal well with misspellings. We use corruptions introduced by the ocr_corruptor to simulate misspellings in the query schema. Headings/Attributes that only contain misspellings are rarely matched correctly, examples where CHAIn fails to produce a correct match include:

- occupled $\overset{CHAIn}{\not\longrightarrow}$ Occupied
- citv $\overset{CHAIn}{\not\longrightarrow}$ City

It is likely that the SPSM algorithm is unable to get any semantic information about these misspellings (since they are not actual words) and is therefore unable to draw conclusions about the similarity between the query attribute and data source heading. To correctly match misspellings, a better approach might be to consider the edit distance[2], which is often considered in spelling correction [14].

4.3.8 Equivalent matches: symbols

CHAIn generally does well to correct mismatches due to the absence of symbols, although this behaviour can also be inconsistent:

- goTerm $\xrightarrow{CHAIn}$ GO_term
- typeOfCourse $\xrightarrow{CHAIn}$ type_of_course
- noOfGraduates $\overset{CHAIn}{\not\longrightarrow}$ no_of_graduates

[2]The edit distance is the minimum no. of operations required to transform one string into the other.

4.3.9 Incompatible matches: null mappings

CHAIn does well at filtering out incompatible matches, null mappings are always returned correctly, showing that results returned by CHAIn are selective.

4.4 Structural mismatches

4.4.1 Order of attributes and missing attributes

CHAIn does well with queries that have attributes in the wrong order, or when queries are missing some attributes. Matches found when running queries with no structural mismatches were also matched correctly when running queries with both semantic and structural mismatches.

4.4.2 Data represented in multiple columns

Different organisations may choose to structure their database differently, for example, an organisation may decide to store Longitude and Latitude values in two columns as floats, while another may choose to store coordinates as a tuple in one column, called LongitudeLatitude. We test CHAIn's ability to correct such structural mismatches with the query *tree(LongitudeLatitude)* to the *tree* data set, which contains columns *Longitude* and *Latitude*. The result is LongitudeLatitude $\xrightarrow{CHAIn}$ Latitude, CHAIn only matches *LongitudeLatitude* to one column, *Latitude*.

4.4.3 Data represented in multiple data sources

Another form of a structural mismatch could occur when organisations choose to store data in multiple data sources instead. We demonstrate CHAIn's ability to correct for such structural mismatches in Figure 4.1.

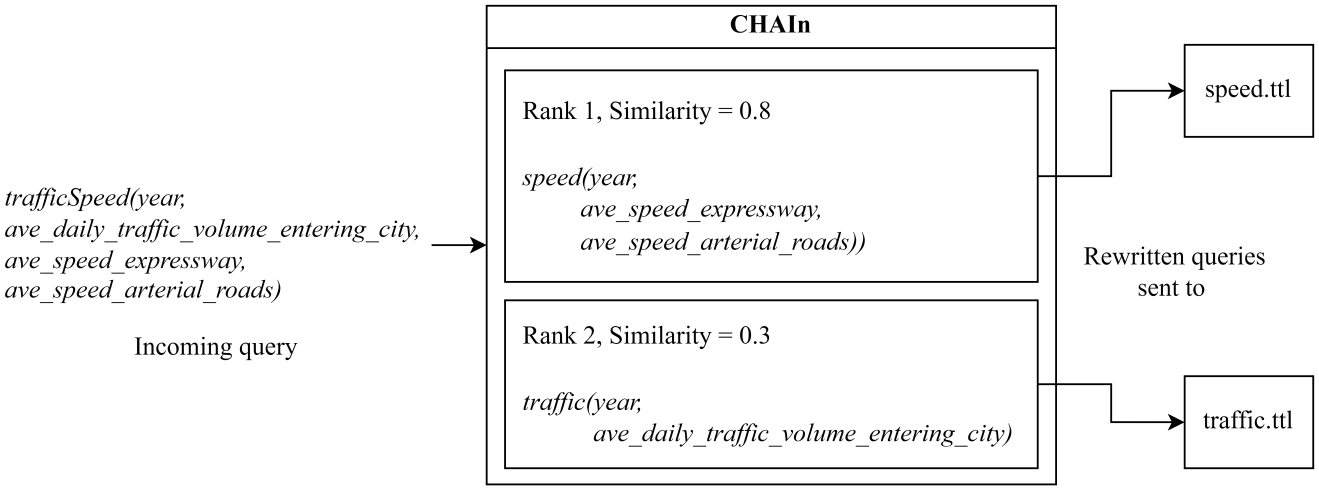

Figure 4.1: Query to evaluate CHAIn on structural mismatches due to data stored in multiple data sources

4.5 Discussion of results

Looking at the results in Table 4.2, matches returned tend to be relevant and precise, although they usually aren't complete, resulting in fairly low Recall. CHAIn's success with null mappings suggests that the results returned are selective and do not contain irrelevant information. CHAIn performs much better on the easier data set than the one which was modified at full difficulty, this suggests that CHAIn might be suitable for correcting minor mistakes in the query schema. If organisations had previously aligned their data sources but later updated their schema, CHAIn is likely to be useful in facilitating information sharing between the organisations. The current implementation of CHAIn still falls short of schema-agnostic querying and is unlikely to succeed if the querying organisation is completely unaware of the target data sources' schema.

Section 4.1 demonstrates how the keyword matching used in the narrow-down process is too limited and causes queries to fail even if there is no mismatch between the query subject and the data source name. A simple solution would be to check if the query subject matches any data source name before starting the narrow-down process. Nonetheless, the process fails to consider many common types of syntax (symbols and digits, proper nouns, and plural forms) in database schemas and needs improvement. Fairly simple fixes such as removing symbols and digits and converting all related terms to lowercase should go a long way towards alleviating this issue.

While CHAIn is fairly good at resolving structural mismatches, results for semantic mismatches are more inconsistent, which suggests that the process of finding matches could be improved. The SPSM algorithm used for finding matches is not selective enough to differentiate between headings that contain the same words and struggles to match attributes that are not actual words whether they are acronyms, abbreviations, or misspellings. Future work could consider using more sophisticated matching systems, using lexical resources that include more information on acronyms, or domain-specific lexical resources instead of WordNet to perform domain-aware semantic matching [16]. Extending CHAIn to also consider the edit distance between the query attribute and data source heading should also help with correcting misspellings and potentially even abbreviations.

Chapter 5

Limitations

CHAIn and crptr are both proof-of-concept systems which present immense potential. However, there are some limitations of both systems that have impacted the scope, methodology and results of this project. We discuss the steep learning curve of working with proof-of-concept systems in Section 5.1, how that led to a narrowed project scope in Section 5.2 and a small sample size in Section 5.3. While not the main focus of this project, crptr is an integral part of the methodology. Hence, we also describe some limitations of crptr in Section 5.4 and describe improvements that could be made to the crptr system.

5.1 Working with proof-of-concept systems

One of the significant challenges of this project was working with two proof-of-concept systems. While both CHAIn and crptr present immense potential for schema-agnostic querying and creating realistic synthetic data for evaluation, actually running the systems is not straightforward. There is a fairly steep learning curve for the user, and significant time and effort were spent setting up the systems. Parameters like the format for input files, the file path for input files, and the metadata required were not included in the documentation and were found after some trial and error.

This project is constrained to black box testing and presentation of results returned by CHAIn since we do not have the requisite knowledge of CHAIn's inner workings to create tests or provide explanations of why CHAIn returns the results that it does. Future work could combine in-depth knowledge of CHAIn's workings and the results collected in this project to implement improvements to CHAIn.

5.2 Narrow scope of project

Partially because much time was spent on setting up the systems, less time was left for the evaluation of CHAIn. The scope of the project is narrowly focused on evaluating CHAIn's ability to find correct matches between the query schema and data source schema. Other aspects of CHAIn that have yet to be thoroughly evaluated.

5.2.1 Data repair process

This project focused exclusively on mismatches at the schema level, but CHAIn is also able to correct for mismatches at the data level. Mismatches at the data level occur when the schemas match but the specific data differ: for example, dates are given as DDMMYY rather than MMDDYY. Future work should also evaluate CHAIn's results when correcting mismatches at the data level.

5.2.2 Ranking of queries

The project envisions CHAIn running as part of an automated query response pipeline, hence we focus on the top result returned by CHAIn (the rewritten query with the highest similarity score). However, this may be shooting for the moon a little too quickly. CHAIn returns a list of all rewritten queries that exceed the threshold, so a more reasonable use case at this stage is for some human user to filter through the list and return the best option to the querying organisation. This has the benefit of providing the user's knowledge and intuition, which is hard to encode in a data set, during matching. Future work at evaluations could consider more queries returned by CHAIn and determine if is indeed more useful with some human input. Future evaluations should also consider if the ranking returned by CHAIn is a good reflection of the similarity between the rewritten query and the data source schema.

5.2.3 Domain-aware CHAIn (DA-CHAIn)

There has been other work done on extending CHAIn such that it uses domain-aware semantic matching to reduce mismatches produced by ambiguity and specialisation [16]. Future work could include a similar evaluation of DA-CHAIn and a comparison of the results collected here against DA-CHAIn's performance. This would help determine if adding domain-aware semantic matching into CHAIn is indeed a worthwhile avenue to develop and improve CHAIn.

5.3 Small sample size

When corrupting a data set, we use the number of modifications and the number of columns are indications of difficulty. However, since the words corrupted are picked at random, and corruption methods are picked with roughly equal probability, these indicators are not consistent benchmarks of difficulty. In the worst case, all corruptions occur within one heading, and all others are unmodified (Figure 5.1).

In contrast, we see another corrupted schema with 5 columns and 5 modifications made by crptr (Figure 5.2). Going by the size of the data set and the number of modifications introduced by crptr, we would expect the schemas to be equally difficult for CHAIn. However, the corrupted schema in Figure 5.1 is likely to be much easier for CHAIn since 4 of 5 headings are identical while the corrupted schema in Figure 5.2 has no identical matches.

Admittedly, the scenario in Figure 5.1 is very unlikely. Given large enough sample

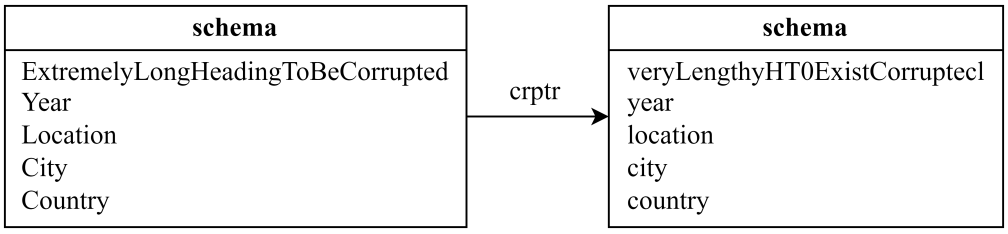

Figure 5.1: Corrupted schema where only one heading is modified.

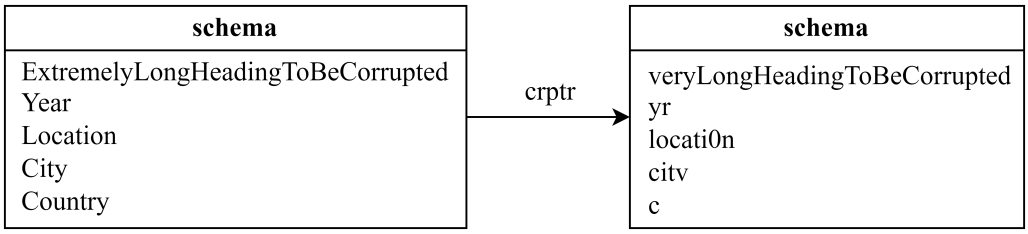

Figure 5.2: Corupted schema where every heading is modified.

sizes, we can identify these edge cases and either remove them or reduce their impact on the results by reporting the average or median results. However, in this project, each data set is only corrupted 3 times at each difficulty level.

Given the time, resource, and expertise constraints, we prioritised testing different data sets across domains instead of using crptr to corrupt a single data set multiple times and running queries. This project focuses mainly on individual matches identified by CHAIn, what sort of mismatches it is good at correcting, and where it falls short. There is less analysis of CHAIn's overall performance since the data is vulnerable to outliers and isn't comprehensive enough to draw conclusions about any trends. Given more time to run more queries, it would be interesting to explore the relationships between CHAIn's overall performance and different data sets, e.g. plotting the relationship between F-measure and the size of the data set.

5.4 crptr limitations

While the main focus of this project is evaluating CHAIn, crptr is an integral part of the methodology used in experiments. Hence, this section will discuss some of the limitations of the crptr system, discuss how it has impacted results, and suggest future work on the crptr system.

5.4.1 Each word is only modified once

As described in Section 3.2, headings are split into individual words, then modified at random by crptr. However, crptr has an issue where it only modifies the first occurrence of each word. Words that are repeated in the schema are only modified once. Even at the maximum number of modifications, crptr produces the schema shown in Table 5.1.

The word "sentiment" is modified in its first appearance in *appleTwitterSentiment*.

Original data set	Corrupted data set
appleTwitterSentiment	appjeChitterView
_unit_id	unjtcl
_golden	g
_unit_state	unitRes_publica
_trusted_judgements	tLegal_opinion
_last_judgement_at	finallyJudgingA
sentiment	sentiment
sentiment_confidence	sentimentC
date	see
id	id
query	inquiry
sentiment_gold	sentimentGo1d
text	t

Table 5.1: Corrupted schema of the appleTwitterSentiment data set

Subsequent occurrences of the word in *sentiment*, *sentiment_confidence*, *sentiment_gold* are left unmodified. We see a similar problem with the word "id". It is modified in its first appearance in *_unit_id* but is unmodified when it appears again in the header *id*.

This behaviour means the data sets with words repeated multiple times are likely to be easier for CHAIn than data sets of a similar size with no repetitions, since some parts of the data set are not modified by crptr. We record the number of columns in a data set, the number of unique words in each data set, and also the presence of repeated words in Section 3.6.

Despite this issue, crptr is still a valuable tool. Its purpose of generating reference mappings to evaluate matching across various domains is fairly novel and has great potential to simplify the task of evaluating matching systems. It should be relatively simple to improve crptr such that it offers the option of modifying every single occurrence of a word.

5.4.2 ocr_corruptor

We use the ocr_corruptor to simulate misspellings in the query schema. ocr_corruptor simulates errors that occur when optical character recognition (OCR) engines are used to digitise records [1]. These errors are likely to occur due to poor handwriting and are based on similarities in character shapes:

- SiteName $\xrightarrow{ocr\_corruptor}$ siteNaiiie
- HeightInMetres $\xrightarrow{ocr\_corruptor}$ hei9htInMetre5

However, these errors are unlikely to occur in databases which do not involve writing or character recognition. Instead, to better reflect errors in database schemas, we should simulate typing mistakes by replacing characters with characters that are close by on a keyboard. Another possible source of misspellings is spelling errors made by

database engineers. Since many misspellings are often due to using some sound-to-letter system [13], crptr could simulate phonetic variations instead, or have a lookup table of commonly misspelled words.

5.4.3 synonym_corruptor

Paraphrasing is more complex than simply picking some related word. One must consider the context where a word appears. Is it acting as a noun or a verb? The scale/degree of the word also matters, e.g. We would rarely use substitute "catastrophe" with "problem" even if they are both unwelcomed situations. Words also could have multiple definitions, so some synonyms may not be suitable in certain situations.

crptr uses the WordNet library to find synonyms, then picks any related term at random without much consideration for the context where it appears. We give a few examples of corruptions generated by crptr's synonym_corruptor that are not appropriate synonyms:

- FilterInstalled $\xrightarrow{synonym\_corruptor}$ separate_outInstalled

  While to "filter" does technically mean to separate out something, in this case, we are using Filter as a noun, while crptr has supplied a verb.

- CommonName $\xrightarrow{synonym\_corruptor}$ plebeianName

  "Common" is an adjective with multiple definitions. crptr has selected "plebeian", a synonym for one of its definitions, i.e. to be from a lower social class. However, CommonName most likely means a name that is often used.

- WardCode $\xrightarrow{synonym\_corruptor}$ Baroness_Jackson_of_LodsworthCode

  At first glance, this appears to be rather random. Understanding that Baroness Jackson of Lodsworth was the respected economist Babara Ward makes crptr's suggestion somewhat understandable. However, it is unlikely that a database engineer would rephrase "ward", meaning an administrative division of a city, with a person with the last name Ward.

It is possible to use domain-specific ontologies and lexicologies with crptr, which would greatly reduce instances of inappropriate synonyms. Ideally, we would have high-quality ontologies for each domain so that synonym_corruptor will select accurate synonyms and perform domain-specific corruption. The current implementation of crptr uses WordNet, which is a general and widely-used lexical resource, but sometimes completely changes the meaning of domain-specific terms, e.g. PubMed[1] $\xrightarrow{synonym\_corruptor}$ taphouseMed.

Domain-specific ontologies are difficult to come by and may be of dubious quality, so crptr will still need to rely on WordNet. Results from WordNet need to be more selective, future work could try to infer more context about the word, e.g. its grammatical category, to find more suitable synonyms .

[1] A database of biomedical and life sciences literature.

5.4.4 Separating phrases

crptr splits headings into individual words before modifying each word individually. Hence, phrases that should not be separated are also split and modified separately. For example, ZIP code is a phrase that should be considered together, zip and code individually mean very different things. A plausible corruption from crptr is ZIP_code $\xrightarrow{crptr}$ zip_upC0de. crptr should be extended to be more flexible such that it can modify phrases while preserving their meanings.

5.4.5 Headings with multiple modifications

Individually, the issues presented by each corrupter function seem acceptable, especially for a proof-of-concept system. However, as mentioned previously, crptr modifies every unique word with some corruption function picked at random. Headers with multiple words undergo more modifications, combining all the modifications can give a corrupted heading that would be hard even for a human to correctly match to the original schema. Table 5.2 shows all the corruptions introduced into one single header: CapitalAssetValueForAmenityTrees $\xrightarrow{crptr}$ Das_KapitalAValveFAgreeablenessTrees

Original word	Modified word	Corruption function
Capital	Das_Kapital	synonym_corruptor
Asset	A	name_abbr
Value	Valve	ocr_corruptor
For	F	name_abbr
Amenity	Agreeableness	synonym_corruptor
Trees	Trees	unmodified

Table 5.2: Example of multiple corruptions in a single heading

An extension to crptr might be to limit the number of corruptions made to each heading, instead of allowing as many modifications as there are words in the heading, which result in drastic modifications that are difficult even for humans to match. Future work suggested in Sections 5.4.2 and 5.4.3 to improve each corruption function should alleviate also this problem.

5.4.6 Discussion of crptr's limitations

While crptr presents a lot of potential for creating realistic synthetic data for query evaluations, its current implementation is not sophisticated enough to create corruptions that are reflective of real-world variations between database schemas. crptr's approach of randomly picking which words to modify and which corruption methods to use, along with the issues where crptr only modifies the first occurrence of a word and applies more modifications to headings with more words means that crptr's behaviour is unpredictable. It is difficult to quantify how difficult the task of matching some corrupted schema to the original schema is. This makes it difficult to compare results across different queries, even for corrupted schemas that were generated by modifying the same data set, and more so for queries querying entirely different data sets.

This project would have benefited greatly from a more sophisticated version of crptr which provides corruptions that better reflects real-world mismatches between schemas. More realistic and reliable results would have enabled us to make stronger conclusions about CHAIn's results. We have highlighted unexpected issues and solutions to introduce more realistic corruptions that are suitable for CHAIn's use case. It should not be too difficult to implement these changes and improve corruptions such that they are more realistic.

A data corruption application that introduces modifications based on requirements has wide applications beyond evaluating rewritten queries. crptr deserves attention and development, as well as thorough evaluation to ensure its utility. CHAIn's unique approach of dynamically rewriting queries using matching makes it difficult to evaluate since gold standards are hard to find or create, crptr is necessary for conducting a thorough evaluation of CHAIn. Given its role in evaluating CHAIn, future work should create a pipeline that connects the two systems, automatically running corrupted queries from crptr and recording the results. This would be extremely useful given that crptr introduces corruptions randomly and requires a large enough sample size to draw strong conclusions about any trends, reducing the need to manually input queries and record results.

Chapter 6

Conclusion

This project has established CHAIn's performance on various types of semantic and structural mismatches. The results on data sets with the maximum number of modifications are inadequate, suggesting that CHAIn is not ready for deployment on data sources where the schema is completely unknown to the querying organisation. The results on the easier data sets show better promise, especially Precision, suggesting that results returned by CHAIn are correct and selective. Unfortunately, Recall could be improved, suggesting that some responses are incomplete (not all query attributes are responded to). This suggests that CHAIn shows promise on data sources on data sources that have undergone pre-alignment but were later updated, resulting in minor mismatches. Furthermore, this project makes the harsh assumption that CHAIn runs as part of a fully-automated query response pipeline. If a human user is involved in the query response process, CHAIn returns the most relevant results, allowing the human user to quickly hone in on the most meaningful results, making the task of sifting through data tractable. Future approaches to extend CHAIn include improving the process of narrowing down the search space for data sources and using more sophisticated matching systems and lexical resources.

There are still other functionalities of CHAIn that should be thoroughly evaluated, including its data repair process and ranking of queries. Future work should also evaluate work done to create a domain-aware CHAIn to determine if that is an effective avenue for improvement.

The evaluation of CHAIn was done using an approximate gold standard matching produced by crptr. Unfortunately, some of the modifications introduced by crptr were not reflective of real-world variations. This restricted the methodology of the project, impacted the results, and prevented us from drawing stronger conclusions about CHAIn's performance. crptr presents immense potential for generating realistic synthetic data, future work could work on extending crptr such that it is more sophisticated and produces more realistic corruptions.

Bibliography

[1] Ahmad Alsadeeqi et al. *Systematically corrupting data to evaluate record linkage techniques*. PhD thesis, Heriot-Watt University, 2020.

[2] Jérôme Euzenat, Pavel Shvaiko, et al. *Ontology matching*, volume 18. Springer, 2007.

[3] André Freitas, Joao C Pereira da Silva, and Edward Curry. On the semantic mapping of schema-agnostic queries: A preliminary study. In *Workshop of the Natural Language Interfaces for the Web of Data (NLIWoD), 13th International Semantic Web Conference (ISWC)*. Citeseer, 2014.

[4] André Freitas and Christina Unger. The schema-agnostic queries (saq-2015) semantic web challenge: task description. In *Semantic Web Evaluation Challenges: Second SemWebEval Challenge at ESWC 2015, Portorož, Slovenia, May 31-June 4, 2015, Revised Selected Papers*, pages 191–198. Springer, 2015.

[5] Raul Garcıa-Castro, D Maynard, H Wache, D Foxvog, and R González-Cabero. D2. 1.4 specification of a methodology, general criteria and benchmark suites for benchmarking ontology tools. *Knowledge Web, December*, 2004.

[6] David George. Understanding structural and semantic heterogeneity in the context of database schema integration. *Journal of the Department of Computing, UCLAN*, 4(1):29–44, 2005.

[7] Fausto Giunchiglia, Fiona McNeill, Mikalai Yatskevich, Juan Pane, Paolo Besana, and Pavel Shvaiko. Approximate structure-preserving semantic matching. In *On the Move to Meaningful Internet Systems: OTM 2008: OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008, Monterrey, Mexico, November 9-14, 2008, Proceedings, Part II*, pages 1217–1234. Springer, 2008.

[8] Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. S-match: an algorithm and an implementation of semantic matching. In *The Semantic Web: Research and Applications: First European Semantic Web Symposium, ESWS 2004 Heraklion, Crete, Greece, May 10-12, 2004. Proceedings 1*, pages 61–75. Springer, 2004.

[9] Roberta Lamb and Elizabeth Davidson. The new computing archipelago: Intranet islands of practice. In *Organizational and Social Perspectives on Information Technology: IFIP TC8 WG8. 2 International Working Conference on the Social and*

Organizational Perspective on Research and Practice in Information Technology June 9–11, 2000, Aalborg, Denmark, pages 255–274. Springer, 2000.

[10] Fiona McNeill, Diana Bental, Alasdair JG Gray, Sabina Jedrzejczyk, and Ahmad Alsadeeqi. Generating corrupted data sources for the evaluation of matching systems. In *14th International Workshop on Ontology Matching*, pages 41–45. CEUR Workshop Proceedings (CEUR-WS. org), 2019.

[11] Fiona McNeill, Andriana Gkaniatsou, and Alan Bundy. Dynamic data sharing for facilitating communication during emergency responses. In *ISCRAM*, 2014.

[12] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[13] Roger Mitton. Spelling checkers, spelling correctors and the misspellings of poor spellers. *Information processing & management*, 23(5):495–505, 1987.

[14] Kemal Oflazer. Spelling correction in agglutinative languages. *arXiv preprint cmp-lg/9410004*, 1994.

[15] Joe Raad and Christophe Cruz. A survey on ontology evaluation methods. In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development, part of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, 2015.

[16] Francisco J Quesada Real, Fiona McNeill, Gábor Bella, and Alan Bundy. Improving dynamic information exchange in emergency response scenarios. In *ISCRAM*, 2017.

[17] Cornelis Joost Van Rijsbergen. *Information Retrieval*. Butterworths, 1979.