University of Twente

Automatic instance-based matching of database schemas of web-harvested product data

Author
Alexander Drechsel

Supervisors
dr. ir. Maurice van Keulen
dr. ir. Dolf Trieschnigg
dr. Doina Bucur

July 12, 2019

Thesis for completion of a Master's degree in computer science at University of Twente

Contents

Abstract

Every day more information becomes available on the internet and companies can significantly benefit from integration of information from these sources that is useful to them into their own systems. However there is no set standard for information on the internet meaning that integrating of this useful information is time consuming and costly. In this thesis we present a semi-automated method for matching web-harvested product database schemas on the basis of data characteristics and commonality. We provide a pre-processing system which takes web-harvested product information and turns it into machine learner ready feature sets as well as a machine learner which is capable of using these feature sets to match groups or columns of data from different sources together on the basis of similarity and thus representing the same property. Multiple methods were developed and tested for sampling, machine learner algorithm and training set selection. We used the best results for each of these. For sampling we concluded that a resampler which generates samples of 100 data values that also has restrictions on how many samples it can generate based on the overall size of the dataset to prevent overtraining performs best. With regard to machine learner algorithms, both nearest neighbour and RbfSVC performed well at the classification task. The system described in the thesis is capable of good matching accuracy scores (in excess of 50% for textual cases and 67% for numerical cases despite a large amount of possible classes) given not many new properties are introduced beyond the training set. The thesis describes a number of clear development ways in which the system could be expanded and further improved.

Chapter 1

Introduction

The general trend is that we register, record and do more and more things digitally. With evermore things happening on a digital level more of their underlying data sources which could contain valuable information are becoming available online. However they lack a standardized structure and individual data source structures can change from day to day. The content within these data sources is of course also continuously changing and expanding.

If one is able to consolidate the content of continuously changing data sources, which contain data they are interested in, into a single data source, and link all data which are about the same entity, in a cost efficient and thus preferably automatic way they could prove a very valuable source of operational or competitive advantage. In our research we will focus on data sources harvested from websites but the developed system should be applicable to any data source containing data which is organized and linked together into items and properties. It should be applicable to all such data sources since in they are comparable in structure and preprocessing will take care of any difference in details that are caused by the format of its contents.

To consolidate different data sources, so they are in a single source and can be compared to each other, you need to determine both how the structures or schemas are similar to one another and once that has been determined, which data records refer to the same object. While studies[1, 2, 3, 4, 5, 6, 7] into automatically determining which records are the same across different data sources (also known as data fusion) as well as into computerized determination of data source schemas (what properties across data sources are identical in concept but are identified using different aliases) are not new research areas they both still remain open for ongoing research. Both topics are a challenge since they have to do with interpreting the meaning of table and attribute names and semantics remains a difficult topic for automation.

For a lot of data integration projects it is still common that the most reliable process is used in which domain experts manually match schemas, determining what properties are basically the same[1] . But due to the labour and thus cost intensive nature of manual schema mapping the number of data sources used

is often limited in those cases. For businesses as well as researchers for whom consolidating data is valuable it would be incredibly valuable to be able to more efficiently automatically consolidate information from big numbers of various data sources including ones with an unknown schema.

In both web harvesting and in the merging of data sources it often occurs that across different data sources data items such as products have properties which while identical in reality, use or concept are registered using different names. Since this concept is key to our research we will clarify that when we are talking about **Properties** we mean data of the same concept across all relevant data sources regardless of the actual name or alias used within those specific data sources. By this we mean data which may be grouped under a different name or alias is meant to mean the same thing in reality or in use. People have a "first name" and a "last name" but these can alternately be referred to as "given name" and "family name" or in different languages such as in dutch a "first name" can be referred to as a "voornaam" and a "last name" can be referred to as "achternaam". These examples can be grouped into 2 properties, a "first name" property which contains the data from "first name", "given name", and "voornaam" and a property "last name" which contains the data from "last name", "family name", and "achternaam". As an example from the test case of our research, which is about ball bearings and which was primarily gathered by harvesting data from webshops, each property of the bearings is described within each data source using a name or alias but the english name of "inner diameter", the dutch name of "binnen diameter" and the domain specific abbreviation of "d" all are referring to the same property. An experienced domain expert would quickly recognize this, but for a computer this would not be easy to learn.

This thesis proposes a method whereby, using machine learning and similarity in data characteristics, properties which are the same in concept are linked together. This method which we will call **Content-Based Property Matching** works on the theory that while properties across different sources may have different names and the data may even be in a different language the "shape" of the data of each property is similar across different data sources. Similar approaches are also employed in [8, 9]. For our research we worked with a case of product data harvested from different webshops about ball bearings and we will use this case to give examples. Bearings themselves are machine components which are used as part of connectors between moving parts to reduce the friction between moving parts and to restrict movement in undesired ways. Bearings are used in a wide variety of applications from dental drills to wheels and gearboxes.

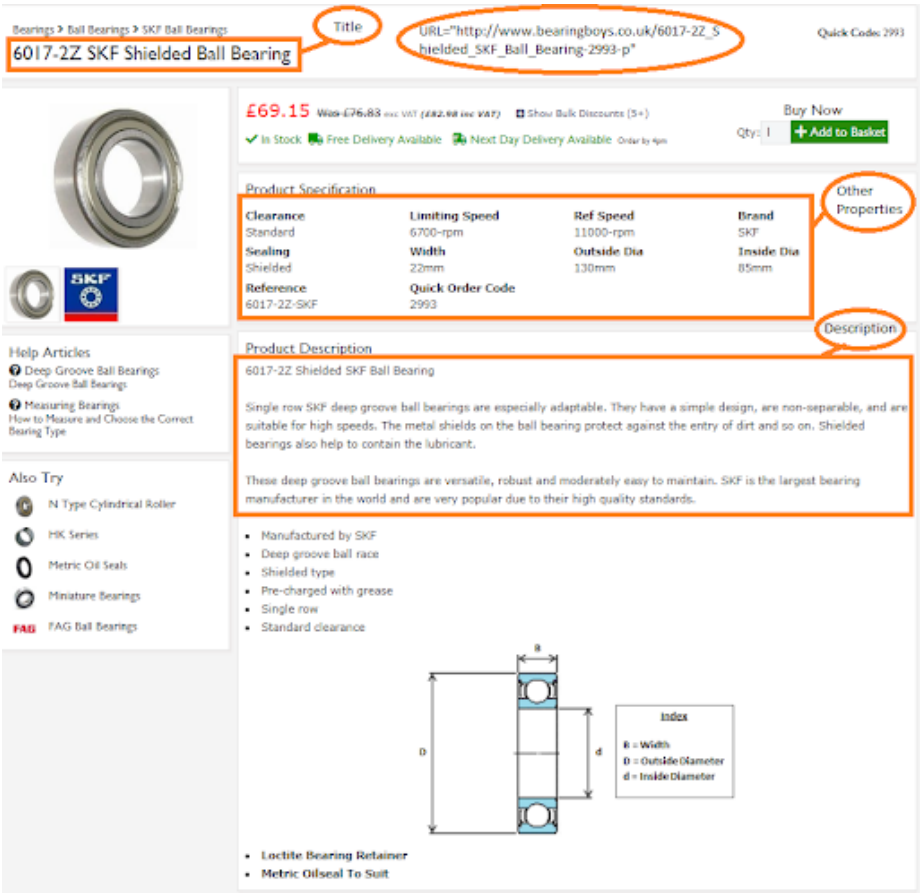

Figure 1.1: A page from the webshop BearingBoys of the 6017-2Z with various properties marked.

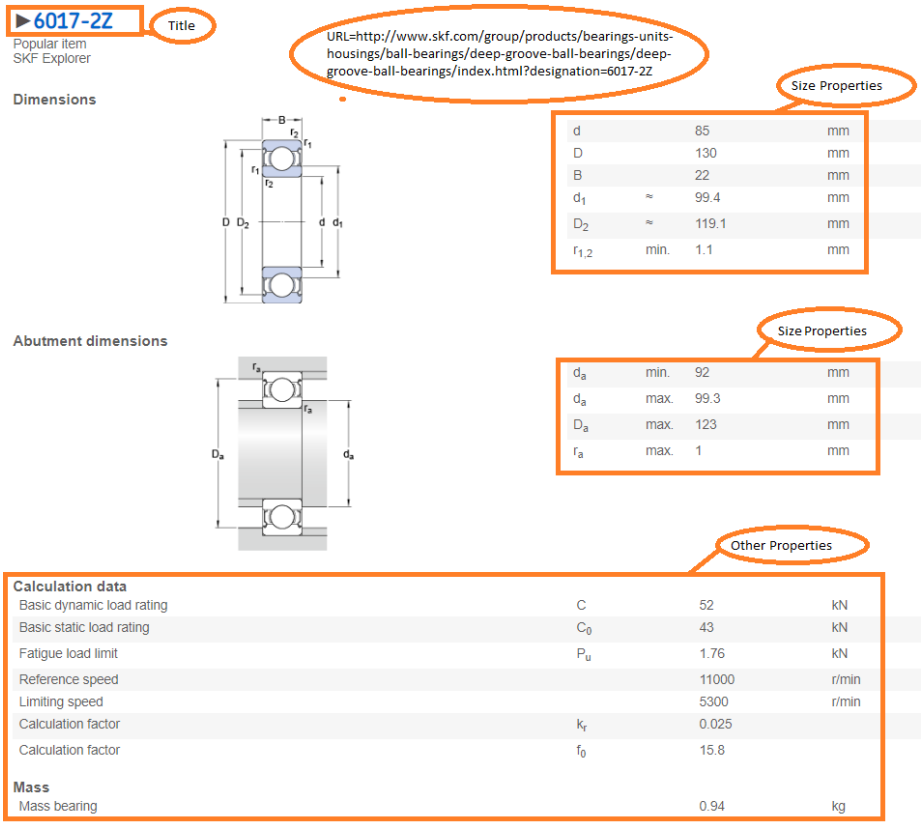

Figure 1.2: A page from the website of the brand SKF of the 6017-2Z with various properties marked.

Presented above are two images of the same bearing from different websites. The two websites share many of the same properties but some properties are still differently named even though they mean the same thing. The product data gathered across websites may also be inconsistent such as in our example images the limiting speed being indicated differently between the websites. Despite this similarities of data characteristics should be detectable for properties across websites. For example generally the product data harvested will contain a title and description describing each product and this will obviously be different across products and websites. However when comparing the shape of the title and description properties they should possess similar shapes across the various data sources. Titles will generally be unique within a data source and each title will be of roughly similar length. These same data characteristics of uniqueness and length should also be similar for descriptions though the length of a description will likely be longer than that of titles.

By using these text similarities we should be able to determine what properties across different data sources are titles and which are descriptions. We

should also be able to use a similar process to automatically determine which numerical properties across different data sources share the same meaning. In the case of numerical values such as various dimensions, weight and product identifiers we should be able to distinguish them from having distinct ranges and other characteristics.

There are a number of different methods by which data characteristics could be made comparable and thus to allow matching based on data similarity for properties. In our case we have chosen to use supervised machine learner based classification. We create our training data as well as process our test data by creating sets of characteristics which in the case of training data are labeled with the correct property. Our characteristics are based on calculations done over the aggregation of data values. Within each datasource we collect all data values that concern the same property. To ensure we have enough values to effectively use machine learning we do not calculate the characteristics based on all data values but instead split these data values up into a number of samples and calculate characteristics based on each of these samples.

Finding similarities to be able to link properties together on an aggregated level would be difficult for a human. Utilizing a machine learner however a computer should be able to handle most of the work and impartially detect similarities. By feeding aggregated characteristics about the data sources into a machine learner it should be able to develop a model capable of predicting to which already existing property new data would most probably belong to on the basis of its aggregated characteristics. Using this model and its predictions it should be possible to determine what properties are referring to the same concept.

Formally speaking the problem we are working on is one of schema matching. That is determining what property in one data source matches with a property in another data source and doing this for all properties involved to be able to determine a fully matching schema. To reduce the amount of work required to achieve fully matched schemas we want to develop a mostly automated method.

Our approach to solving this problem is to utilize the similarity of data values of properties and determine schema matching on the basis of that.

1.1 Research Questions

1.1.1 Main Research Question

1. How can we link properties which represent the same real world property or concept together, using the characteristics and commonalities of their data.

1.1.2 Sub-Research Questions

1. What are existing processes for linking properties together in the use of database merging and web harvesting? Both automatic and manual.

2. How can a database or harvested web page be made suitable for linking properties together?

3. How can we best configure a machine learner and features representing data characteristics to perform content based property matching?

1.2 Validation and Approach

We intend to validate our research experimentally. By using our process to develop a machine learner model and to evaluate the predictions of this model. We will use a collection of data sources created from a web harvesting process. We will evaluate the results of this on both an individual level, where we examine why certain properties seem to match better or worse than others, and an overall level in how it performs overall. We evaluate this mainly by accuracy.

Chapter 2

Related Work

As the introduction discussed this research is about the linking of product properties by the use of a machine learner and features built on data commonalities of the properties. In this section we will discuss the following:

- DIKW Pyramid: To explain the relation between data, information and knowledge.
- Machine Learning: As that is our primary method of solving our problem.
- Data Integration: To explain how our problem is placed in a greater whole.
- Record Linking: As the techniques used here can be made applicable to our problem
- Schema Matching: Which is what our problem is.

2.1 DIKW Pyramid

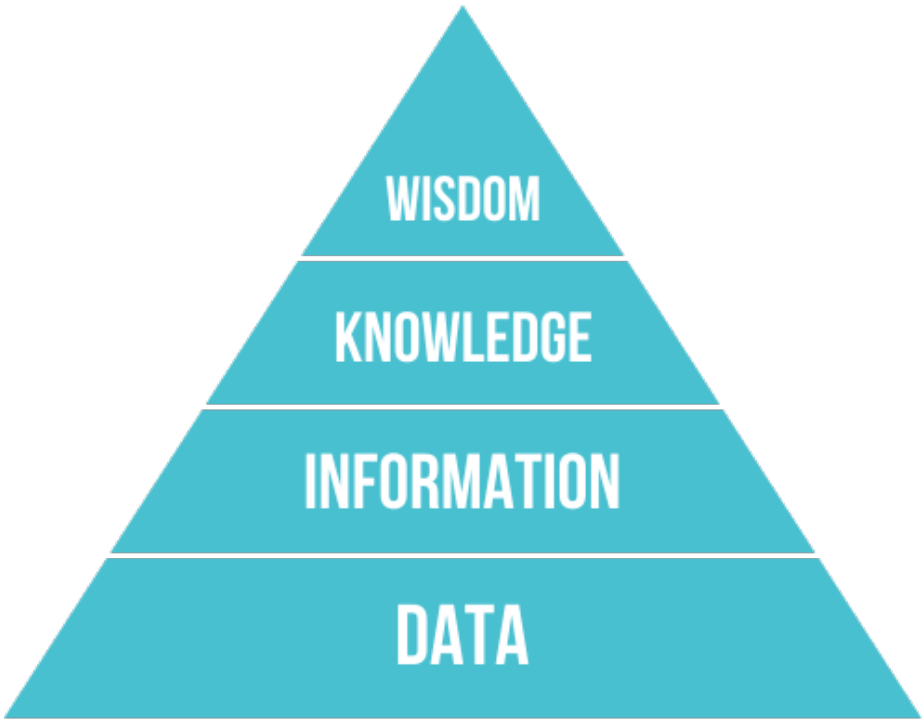

Figure 2.1: DIKW Pyramid

When working with data with the aim of processing it in some manner to gain more information about it the concept of the DIKW pyramid is often used be it consciously or unconsciously[10]. Within this thesis we do not explicitly use the DIKW pyramid but we do use it implicitly, treating data, information, and knowledge as different levels. The DIKW pyramid is used to describe the relation and definition of data, information, knowledge and wisdom. Data is the lowest level of the pyramid and pertains to the recorded values. Without any context data is of no use. The second level of the pyramid is information and is generally achieved when data becomes useful. Generally this is by providing context to data be it by description, structure or another way of making data meaningful or purposeful. The third and fourth levels of the pyramid, Knowledge and Wisdom are considered harder to define. A definition of knowledge is the insight, understanding and experience gained by working with contextualized information. It is often tacit but embedded in the doing within its domain (and outside it). Wisdom is not always included in DIKW and sometimes dismissed but it could be considered knowledge about knowledge. For our research we desire to combine the structural information of data sources together using

the similarities in their underlying data. In this context the structural information could also be considered data and by processing it with our methodology we can produce contextually useful information. In other terms we explore the data for commonalities and similar patterns to uncover hidden information. This hidden information can then be used to structurally match data sources. By using the process and working with the data and information, we gain knowledge.

2.2 Machine Learning

Machine Learning is a broad field within computer science with many applications pertaining to acquiring information from data and is capable of "learning" as the amount of suitable data increases. The concept of machine learning was postulated all the way back in 1950 by A.Turing [11] though the actual term was coined in 1959[12]. Since this research is an application of machine learning and there are multiple good books and papers available [13][14][15], this section will merely give a brief summary and how it is relevant for our research, referring interested readers to the referenced material and other information available.

There are multiple ways to categorise types of machine learning such as categorising based on its learning system and based on its desired output.

When categorising based on learning system or feedback signal there are two broad categories: Supervised Learning where the system is provided with a training set of example inputs and the goal is to learn of to match new data to the example data (in the form of the target outputs found therein). Within this category there are a few subcategories, Semi-Supervised Learning where the target outputs in the example data may be incomplete, Active Learning where the algorithm can query other data sources and the user for feedback during the learning in a limited manner and Reinforcement Learning wherein the training data is generated from a dynamic environment. The other category is Unsupervised Learning where the training data is not given any labels and he system is left to find structure and patterns on its own. Finding such patterns may even be the goal itself in such systems.

A different way of categorization is to base it on the desired output. In general machine learning tasks have similar desired output, which is the grouping or finding patterns in data. There are however a few distinct forms of output within this.

- Classification: Classifying inputs into distinct classes. Usually this is done using a learned model which has been built using labeled training data. Determining whether e-mail is in the class "spam" or "not spam" is an example of this.

- Regression: Similar to classification only the output is continuous, so instead of distinct separate classes it is assignment on a scale.

- Clustering: Forming the input data into groups. The advantage of clustering is that it does not use predefined groups making it well suited for

handling unlabeled data. Due to this it is often used to handle unsupervised learning tasks.

- Density Estimation: Used for estimated how a larger dataset may be distributed from a smaller sample.
- Dimensionality Reduction: A method to separate data into lower dimensionalities such as separating the contents of a library on the basis of topics.

The task we are working on for this thesis is the matching of database schemas. To accomplish this task we plan to match the different properties from databases on the basis of data commonality using supervised classification. After some preprocessing each database will have a number of feature sets created from each property. These feature sets represent the data in a machine learner processable format which can be used for the comparison and matching on the basis of data similarity. We also have manually created class labels for each property so that the correct match is known for the purposes of training. By taking these inputs we perform supervised classification with machine learning. Our desired output is then the feature sets classified to belong to global properties. Combining this classification together with the link from which database level property each feature set was created from allows us to generate a database schema match for each used database to a global schema. The trained machine learner can then also be used to classify new databases to generate schema matches without the need to manually label the new database.

2.3 Data Integration

Data integration is the combination of different data sources about the same subjects to gain a consolidated and concise view of the data. Broadly speaking there are two methods by which data integration can be achieved.A <G,S,M>method such as described in [3] where the databases are not merged in actuality but has 3 elements allowing it to create unified views and process queries.

- A global database schema (G)
- Individual database schemas (S)
- Mappings (M) which define how to transform queries and results between the global database schema and each individual database schema.

While the individual database schemas should be known, the database matching and mapping required to create a global database schema and the required mapping generally need to be created. In this case for each individual query there is a processing step applying the necessary transformation for each database. The second method is more costly to implement but merges the databases in actuality. This data integration method such as described in [1] has 3 steps:

1. Database schema matching and mapping, similar to the <G,S,M> method this is determining a global schema except the mapping is used to add all data from the individual databases into the global database instead of how to transform queries when they are called.

2. Duplicate Detection, determining which records refer to the same real world entities

3. Data fusion or data merging, merging the duplicates found together and dealing with the inconsistencies this generates.

This thesis focuses on providing an automated machine learning based method for the database schema matching step, primarily with the second methodology of actual database merging in mind but is applicable to the schema matching required for both methods of data integration.

2.4 Record Linking

While this thesis is interested in linking properties in an effort to provide automation in schema matching, the techniques used in record linking for "row-by-row matching are also relevant and applicable to our concept for data-driven schema matching. We make these techniques applicable by treating the properties or columns of each data source as records. Thereby allowing us to match "column-by-column". The data contained in each column treated as the data which composes a record. Record linking is the task of linking records which refer to the same entity between multiple data sources. By linking records more complete information about entities can be built up. Record Linking can be considered a header for a field of linking methods and applications such as duplicate detection (Often seen as part of a data integration process as described in 2.3), entity linking and entity resolution. There is a wide variety of methods by which record linking can be done and the methods are often used in conjunction. The following list explains a number of these methods.

- Standardization, as data can come from many different sources it is common that data that contains the same information is often represented in different ways. Standardization or data preprocessing[16] works by ensuring all data which represents the same property is represented in the same way. Thus for example ensuring that dates are ordered in the same order and constantly use the same numerical or textual representation (2st of january 2019, 02/01/2019, 2 jan 2019, and 01/02/19 all being standardized to the same representation) or the same unit of measurement is used (lengths of 5 cm, 0,05 m, and 1.9685 in). Standardization can be achieved in a number of ways ranging from string replacement, tokenized replacement, or more complex methods such as hidden Markov Models[17].

- Rule-based record linkage is one of the more simple methods where records are linked on the basis of one or more key identifiers[16], potentially with a

threshold based on how many identifiers need to match. A similar method is also used internally by relational database which link their records together using specifically defined primary keys.

- Probabilistic record linkage: while rule-based record linkage often looks for exact comparisons, by using probabilistic or fuzzy linking methods it is possible to roughly predict which records should be linked to each other. There is a variety of ways by which this fuzzy linking could be achieved[2] ranging from the simple addition of a edit distance to string comparison to the implementation of a Naive Bayes algorithm.
- Machine Learning, the general concepts of the above methods can also be enhanced and expanded by the application of machine learning techniques as well as opening other options.

2.5 Schema Matching

When wanting to combine multiple data sources (such as through a process as described in 2.3) whether it be for the purposes of merging or wanting to view and compare them with each other, it is important to know which elements or properties of each data schema are semantically similar or identical. Schema matching is the task where these semantic links are determined and it is an important and early phase to when combining data sources. While most of these semantic links will be a one-to-one relation, where one data entry is comparable between data sources, this is not necessarily the case and more complex relations such as a one-to-many link are possible. A common example is a name which in one source is represented via the name property (which contains both a person's first name and last name) and the other source which represents a person's name using two separate properties, firstName and lastName. While schema matching is still commonly done manually, systems which are capable of (semi-)automatically performing schema matching do exist. A broad variety of techniques and approaches to perform schema matching exist, [7] provides us which a large list of approaches. While this list is complete in terms of techniques it lacks the distinction that instance based matching is not a specific technique but many of the approaches can be applied either on a schema level or on an instance level. This distinction is clearly illustrated in [6] though this source lacks more recent approaches such as matching based on usage statistics.

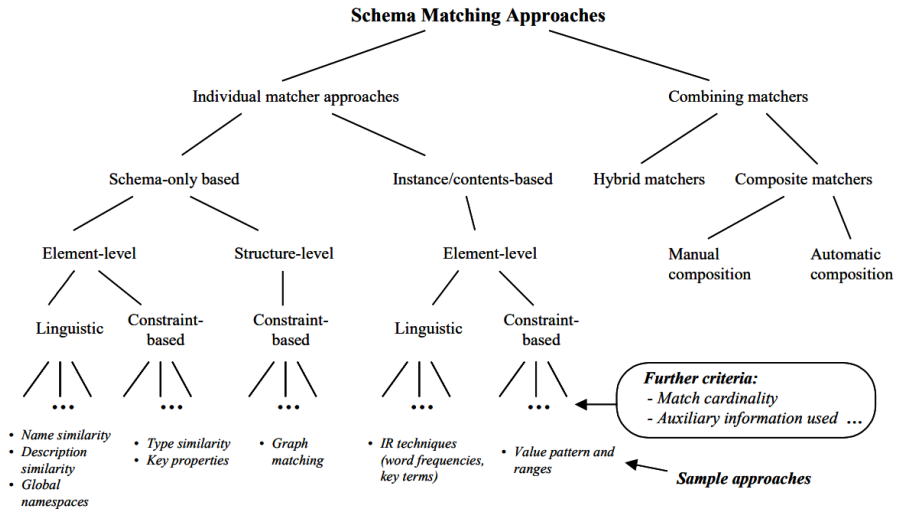

Figure 2.2: Tree Graphs from [6] which classifies schema matching approaches. The method proposed in this thesis is a instance based method using both linguistic and constraint approaches.

Regardless of how these approaches are categorized however it is important to note that what is explained in these sources are general approaches and not specific techniques on how to implement these approaches which can potentially be done via simple string comparisons or more complex methods such as using machine learning. Schema matching systems often combine these approaches in different ways using different implementations. As an example for how schema matching systems combine different approaches and implementations together: [5] describes a schema matching system called LSD which is used to integrate multiple data sources about real estate together. At its essence LSD uses 5 different techniques to match schemas together:

- Name Matcher which is a schema level linguistic matching approach implemented using a nearest neighbour based classifier called whirl[18] which has been developed specifically for text.
- Content Matcher which is a instance level linguistic matching approach based also implemented using whirl.
- Naive Bayes Learner is also an instance level linguistic matching approach except now implemented using a Naive Bayes learner where each instance is treated as a bag of tokens.
- County-Name Recognizer which uses auxiliary information to specifically recognise if a property is a county name.
- Constraint Handler which is a schema level constraint based approach.

When considering what schema matching approaches to use it is important to realize that different approaches requires different information which may not always be available.

- Constraints based approaches require that those constraints are known to the schema matcher.
- Using auxiliary information requires that such auxiliary information is available in a usable format.
- Matching based on usage requires that usage logs or some similar data is available.
- instance based approaches require that a sufficient number of instances is available to actually be able to use them to make determinations about the whole schema rather than just coincidental links.

Our problem is the schema matching of web-harvested product data and due to this we lack a lot of certainty about the underlying databases and our approach has been designed to limit what additional information is required. We do not know the constraints set on each datasource nor do we posses any usage logs. As we are working with a single product domain of ball bearings we could use auxiliary information but do not do so beyond generic units of measurement in an effort to limit additional required information. We do however require a sufficient number of instance data.

Chapter 3

Approach

To merge separate data sources we need to find out how and where these sources fit together. This is a difficult task with multiple steps required. Our research attempts to provide a mostly automated method for the schema matching step of this process. We do this by using machine learning methods to match columns of tables or their equivalents based on the shape of their data.

Formally our task is to: Classify the properties of each data source such that those properties that are the same in concept are classified together. The classification should be based on the characteristics of the data of each property. To evaluate the success of our task we can exclude part of our data during the training phase of the classifier and evaluate the classifier using the accuracy scores of the predictions by the classifier for this excluded part of the data.

The dataset we are working with for this thesis consists of 9 data sources which contain product data about ball bearings. The data sources are: bearingsonline, bearingsdirect, motionindustries, xbearings, rfc, bearingboys, btshop, eriks, and abf. These data sources are JSON files which have been created by a web scraper. Each JSON file has a list of elements. Each of these elements have 3 components: A harvestID which is generated by the harvest for each individual webpage, a property name, and a data value. These elements are generated for each property the web harvester detects on each page. In addition to the datasets we manually created a match file where the properties of each file have been manually classified. We use this match file as our ground truth.

3.1 Processing Systems

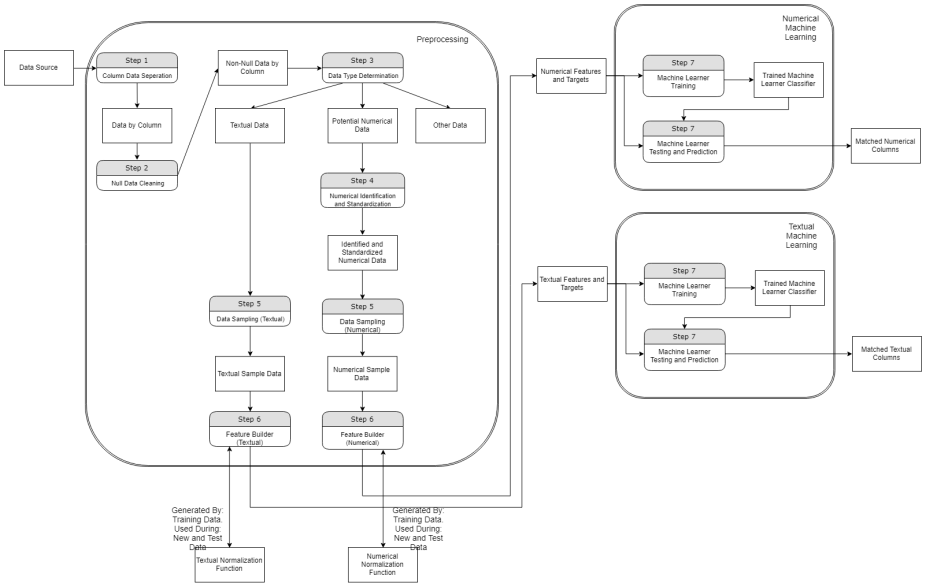

Figure 3.1: The flow diagram of the full design of the system and through what steps data go and are transformed into.

The process is divided into multiple steps:
Pre-Processing Phase

Step 1. (Data Homogenization) Separate and transform the data sources into a homogeneous form. Since we are interested in specifically the properties of the data and want to determine how they are linked together based on similarity in shape we do not need to preserve all of the existing structure but instead only the core of what we are interested in. Since we are potentially dealing with multiple heterogeneous data sources of differing formats we can have multiple data transformation systems each of which designed to deal with a different data format. The end results for each of these systems will be the same: A series of headers each with a list of their associated data. We refer to such a data structure as a column.

Step 2. (Null Data Cleaning) Removal of all null and empty Data. This is important since our test cases are using data harvested from the internet so we cannot expect 100% clean data and it is a given that this should be assumed to be the case for most real-life databases. Null and empty data are not relevant for our matching process and would only hinder our matching or cause errors.

Step 3. (Data Type Determination) Determination of what is exactly the type

of data for each property or column of data. Primarily this would split the properties into textual and numerical properties though other properties such as dates and multiple properties concatenated together (such as Length x Width x Height) could be handled in a different manner and could be considered part of a different category than numerical and textual, but we considered that outside our scope.

Step 4. (Numerical Identification and Standardization) Identification of the exact part of our numerical data that is actually the numeric value. This step contains 2 sub steps related to dealing with numerical values and their origin from different sources.

   (a) Ensure that decimal and thousands markers are interpreted correctly. This is important since differing countries can use different characters to denote these markers.

   (b) Identification and standardization of the units of measurements. This identification and standardization step is quite key since without it properties of the same type would not be fully comparable since characteristics such as ranges and averages would be calculated incorrectly.

Step 5. (Data Sampling) To ensure we have the multiple data points needed for machine learning-based classification we divide our columns up into samples of smaller size.

Step 6. (Feature Building) Calculation of sets of features from the samples of the now cleaned and standardized data. This step contains 2 substeps applying final data transformations related to the use of machine learning.

   (a) Normalization of calculated features using the Unit Standard Deviation technique. We do this to reduce the effects of abnormalities and to ensure that all features are evaluated equally.

   (b) Apply our manually created matches to the associated headers to add our desired target answers for later training and testing purposes.

Machine Learning Phase

Step 7. (Machine Learner) Train the machine learner to form a classifier using part of the feature sets and target answers we calculated in the preprocessing phase.

3.2 Step 1. Data Homogenization

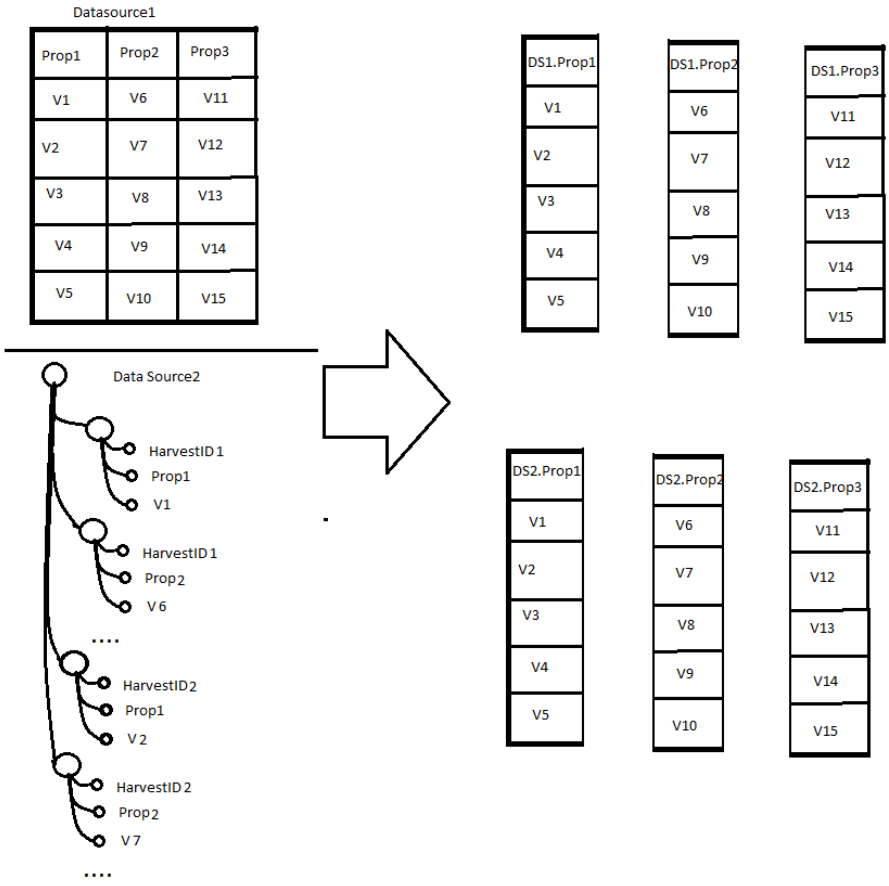

Figure 3.2: A visual representation of Data Homogenization.

In the Data Homogenization step we transform heterogeneous input into a homogenous structured output to be used within the rest of the process. To do this we would have several systems which each handle one distinct type of input with the goal of each of these systems to generate the same output.

Within this step we take information from any data source in whatever format it is and aim to transform it into separate lists of data (each headed by their own property name) that each of the values within that list belong to. We are not yet aiming to match property names across data sources in this step so we only consider values attached to property names when the name of the property is the same within its own data source and assign an additional data source related identifier to each resultant list to keep them distinct between

different data sources as well as to preserve potential hierarchies within the data source. Within the context of this paper we refer to this structure of a list of data values from a single data source headed by the property name they are associated with concatenated with its data source and hierarchy within that data source as a column.

Depending on the exact object type format of the data source, different systems are used. For relational databases the desired output is achieved by taking each table as an individual datasource and separating their columns into lists of data values, each headed by a column header which includes the full hierarchy of this column, meaning the data source, table name and column header concatenated together. Within our testcase of harvested ball bearing data from template-based websites such as webshops we have our data provided in the form of JSON files which have a structure as displayed in Figure 3.2 as Data Source2. For this type of data we load the JSON file as a list of lists and work downwards from the top adding each value to a new list of its matching property name, creating new lists when new property names are encountered. Similar systems would be created for any other format of data sources.

3.3 Step 2. Null Data Cleaning

Once all data has been translated into a universal format usable for our pre-processing steps we need to do some data cleansing. While later additional data cleaning may be required depending on what data type each part of the data set is determined to, be we can already do some cleaning at this point of unusable values. Specifically we can remove null and empty values, since those would only hinder the matching process or cause errors. Most of these values will be missing data rather than actual empty data that actually has meaning. To accomplish this, we simple run a checker over each column which removes unusable values.

3.4 Step 3. Data Type Determination

The next step in the process of turning the data we have into a usable form for our machine learner is to determine what type the data has, since depending on the data type we need to treat it differently. Within our research we decided on two main data types but other specialized data types are imaginable. In theory different data types should not overlap and will mostly require slightly different processing thus in all following processing each data type is treated separately including in the final machine learning classification steps. The first type of data we have is textual which can also be seen as the default data type and is considered difficult to look at the content of the data, since natural language processing is difficult especially for comparison purposes when considering multiple possible languages. Our second main type is that of numerical data which is one for which comparison should be significantly easier. Other data types

could include types such as datetimes where comparing should be done differently due to the way days and months roll over and combined data types which include multiple values such as a dimensions property which includes height, width and length.

In theory distinguishing between textual and numerical data should be easy especially if we consider the textual data type to be default and we only really need to determine if data is numerical. However there are several factors which make it more difficult in practice. Firstly one of the intended purposes of this method and our research case is to enable the merging of web harvested data which mean that we need to be able to deal with dirty data or other forms of values where a numerical value may not purely consist of numerical characters. There are two levels where we need to consider when determining if we should consider data to be numerical. First we need to determine if a specific value is numerical and a column as a whole is numerical or otherwise.

On the level of columns we decided to use a percentage threshold which needs to be reached by individual values considered to be numerical. If this threshold is reached than all individual values that were not determined to be numerical will be discarded. We decided to discard these values since if the threshold is reached we are most likely dealing with a numerical based column in which case the non-numerical values would not be very useful since we are keeping the data types seperate and using those remaining values as textual data would result in small columns of data of insufficient size for productive matching.

When determining individual values it is important to consider a numerical value does not consist purely of numerical characters but that in the end we would prefer to only end up with numerical values. This means that we need to make allowances for values containing units of measurement, dots, commas, odd formats such as fractions and other additional text. At the same time however there will be instances where we do not want to consider something to be numerical simple because it contains numeric characters. An example of such an instance would be alphanumeric codes (such as item codes) and compound values. While there are multiple approaches possible to solve these issues. The approach used in this research is a specialized regex or regular expression. Using the regex we account for the various difficulties but the regex expressions have been hand-crafted on the specific dataset and a better more generic solution should be possible. A good example of this specialized nature is the inclusion of see diagram image as that present due to some data sources containing numerical values where this section of text is present.

The regex expression we used for the data type determination is as follows:
"[0-9]+[.]?[0-9]*[\-]*([0-9]+/[0-9]+—/[0-9]+)?[\-]*(rpm|mm|inch|in|degrees c.|degrees f.||each|g|n|kn|lbf|kgf|lb|lbs)?[ ]*(see diagram image)?"

The regex expression is dividablable into a number of sections.

- [0-9]+[.]?[0-9]*[\-]*([0-9]+/[0-9]+|/[0-9]+) identifies the actual number
- [\-]* Scans for spacing characters

- (rpm|mm|inch|in|degrees c.|degrees f.||each|g|n|kn|lbf|kgf|lb|lbs)? is an enumeration of units of measurement
- [ ]* spacing characters
- (see diagram image)? Section of text which occurs within some data sources contained within some of their numerical values

This regex expression is meant to identify numerical data within our dataset by full matching their data value. This also includes any text within the same data value. Units of measurement included in these data values are useful for identification and standardization performed in step 4. see diagram image is not useful in step 4 but is used in the matching process for completeness since it is present within the data and its inclusion allows us to use full matching instead of partial matching. Reducing the chance of incorrectly recognising a textual value as a numerical one because it happens to include a number. Due use of this regex it is likely that when new data sources are added this regex will need to be extended due to new units of measurements or other text included in numerically typed data.

3.5 Step 4. Numerical Identification and Standardization (Numerical Data only)

While extracting features beyond the shape of the data is difficult for textual data due to the difficulty in processing natural language this should not be the case for numerical data. If we want to extract additional features from numerical data however we do need to determine the actual number contained in each numerical value. To achieve this we need to take a number of steps. According to current standards[19] both dots and commas can function as the decimal separator and no thousands marker should be used this may not necessarily be the case for the data we are working with. The programing language we used considers a dot to be a decimal marker with commas having a different use. Thus we need to remove any possible thousands markers and ensure all comma decimal separators are replaced by dots.

Once we have accounted for the use of decimal and thousands separators we should now determine what part of the data values we are checking is the actual numeric value. Within our datasets we encountered three ways the actual numeric values were displayed.

- A number potentially with a decimal separator.
  - 70.5 is an example of this
- A number displayed as a fraction.
  - 2/3 is an example of this
- A number followed by a fraction

  - 2 1/3 is an example of this

To identify the actual numerical value in each of these cases we once again use regular expressions.

- Our data values still need to satisfy the regex used in the type determination process so it should still satisfy "[0-9]+[.]?[0-9]*[\-]*([0-9]+/[0-9]+—/[0-9]+)?[\-]*(rpm|mm|inch|in|degrees c.|degrees f.||each|g|n|kn|lbf|kgf|lb|lbs)?[ ]*(see diagram image)?"
- We check to see how many matches with '[0-9]+[.]?[0-9]*' we have.
  - If we have a single match this means we are dealing with the first case and can simply change the string value to numerical value.
  - If we have two matches and '[0-9]+[.]?[0-9]*[\-]*\/[\-]*[0-9]+[.]?[0-9]*' can be matched we are dealing with the second case. To acquire our actual numerical value we use division with our two matches to acquire our a decimal representation of our fractional
  - If we have three matches and '[0-9]+[.]?[0-9]*[\-]*[0-9]+\/[0-9]+' can be matched we are dealing with our third case. To acquire our actual numerical value we use add our first match to the division with the second and third matches to acquire our a decimal representation of our fractional

Once we have identified the actual numeric value we could consider this step a success but to aid in comparison we would also like to ensure that values of the same physical quality (Lengths, Weights and Force to give some examples) are all using the same unit of measurement. For our test case we simply have a sequence of if/else statements checking for the presence of units of measurements and standardizing them towards the most common unit of measurement in their category. We standardize towards the most common unit of measurement instead of the SI standard since not all numerical values have units of measurement associated with them so converting to the most common should give the best results for comparison. The methods we are using here are somewhat simplistic and more complicated but better performing methods are possible.

3.6 Step 5. Data Sampling

To be able to properly use our data in machine learning for the purposes of classification we need to turn our data values into features of those values . However we need a sufficient number of such feature data for machine learning and not merely a single set of features per property. To have a sufficient number of feature data per property which each still is sufficiently characteristic enough about the property we propose to split the data we have up into samples.

In our Data Sampling step we break up the columns we have into samples. During this research we developed multiple methods by which samples could

be created. In the experiment section we will evaluate which of these methods performs best.

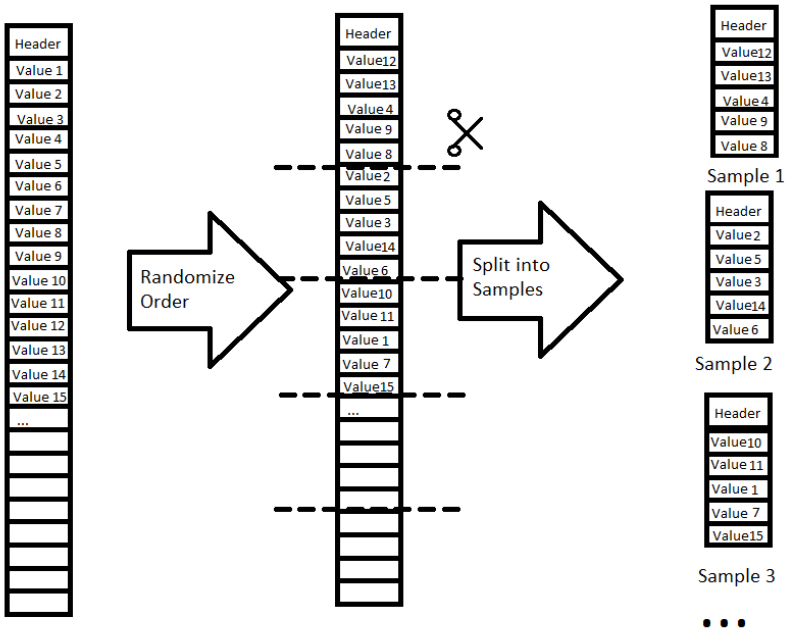

Figure 3.3: Data Split

Our first method **Data Split** is to simply divide the columns up into samples of the desired size. We select these values randomly instead of in order to get a good average of the data and to avoid any potential ordering which was present in the original input. A column will only coincidentally have a number of values divisible by our sample size. Columns will generally not be perfectly divisible by the sample size thus we will generally be left with a remainder. We have two options within our process. We do not use it regardless of size or we use it if the remainder is at least half the size of a normal sample.While this method works fairly well on large columns it is not ideal for smaller columns which may not generate enough samples for the machine learner to use by simple splitting.

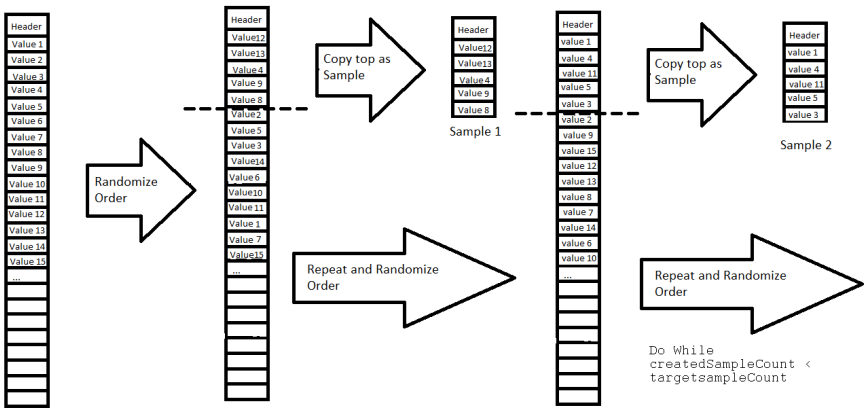

Figure 3.4: Resamping

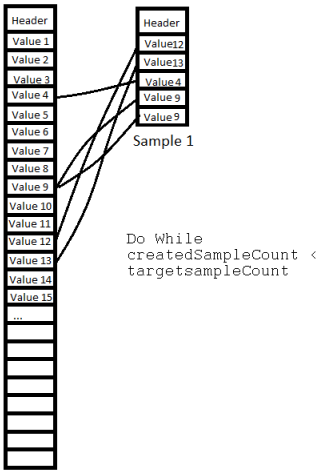

Figure 3.5: Resamping

As an alternative we developed various methods which can resample the data, meaning that it is possible to extract more samples from a column than would be possible from splitting the column. In general the resampling methods work by randomly selecting values from the column to use in creating samples and keeping the selected values selectable. We developed two resampling methods called **Resampling** and **Individual Resampling**. The differences between the resampling methods are in when values become reselectable. In Resampling values only become reselectable once a sample has been created, meaning that

within the same sample the same exact value entry cannot occur twice. In Individual Resampling a value never becomes unselectable, meaning that while extremely unlikely it is possible for a sample to be created from a single value entry repeated.

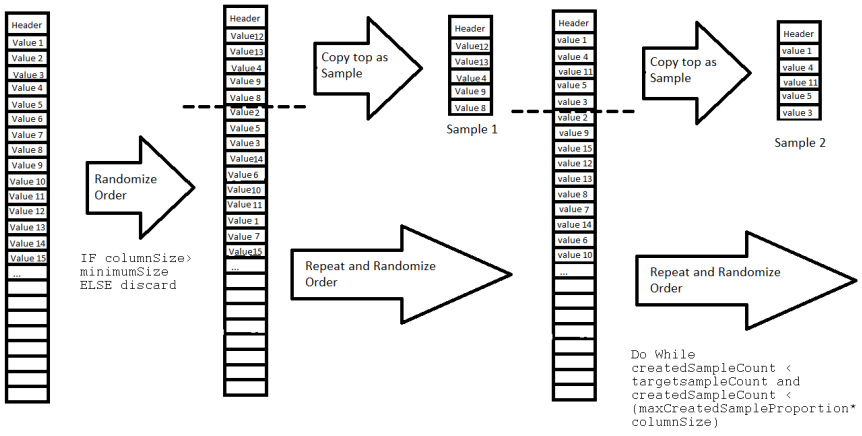

Figure 3.6: Resamping with Limiters

A final refinement added into the resampling creation process is the addition of limiters, clearly defining a minimum size a column needs to use it for the sampling process. and maximum limit in how many samples may be created from a column based on the proportional difference between the size of the column and the size of a sample. We called this method **Resampling with Limiters**.

We evaluate the performance of each of these sampling methods in section 4.1.

Since the goal is matching columns together, each created sample also has its original column and data source name associated with it so that it is traceable.

3.7 Step 6 Feature Building

With the various columns divided into samples the Feature Building step turns those samples into a format understandable to machine learners. By calculating various features for each sample we get numerical representations of the shape of the columns. The created features differ between the various data types since both what we have to work with and how they differ within their data type is different.

For numerical data we have the actual numbers to work with and expect the shape of numerical columns to be found in the form of ranges or in the repeat of data. Based on this we calculate six features for numerical data.

- The value of the most common entry within the sample
- The number of unique entries represented as a percentage to avoid the sample size from influencing this.
- The minimum numerical value within the sample
- The maximum numerical value within the sample
- The mean of all numerical values within the sample
- Standard Deviation within the sample

In the case of textual data part of the problem is that the machine learning algorithms that we use cannot utilize text directly, there are multiple ways to deal with this problem such as a representing it in numbers using a bag of words or other methods. For our purposes however we are trying to match based on the shape of the data and are trying to sidestep language issues because of this we instead calculate most features based on the character and word length of the data. For textual features we generate the following:

- The number of unique entries represented as a percentage to avoid the sample size from influencing this.
- The minimum amount of characters in a value within the sample
- The maximum amount of characters in a value within the sample
- The mean amount of characters within the sample
- Standard Deviation within the sample in terms of amount of characters
- The minimum amount of words in a value within the sample
- The maximum amount of words in a value within the sample
- The mean amount of words within the sample
- Standard Deviation within the sample in terms of amount of words

Once the features have been built we have two transformation steps to perform as well.

Firstly, as mentioned in the section about textual data, machine learning algorithms do no compare text directly this also counts for the associated original column and data source name which essentially form the target answer which the machine learner needs classify to. Specifically we want the target answer to be the attribute we manually matched to the column meaning that the target answer is the same across all the same columns from each data source. For this we have a transformation table which we refer to transforming each original column and data source name to a number we associated with the correct attribute type and if we manually determined columns to contain the same type

of attribute the number transformed to will be the same. For example we have the column called Inner Diameter (d) from the data source xbearings referring to our manual match file this is associated with the attribute Inside Diameter and referred to by the number 4. The numbers used to refer to attributes are generated based on line the attribute is listed in the manual match file.

Secondly, we perform normalization on the generated features. We perform Unit Standard Deviation Normalization which means we express each feature in the form of the number of standard deviations it is away from the average of that feature across all features of its type. For example consider an inner diameter attribute which has a mean of 30, and a standard deviation of 5. By applying Unit Standard Deviation Normalization we express values into number of standard deviations away from the mean instead. In our example this means that a value of 20 is normalized into -2 and a value of 35 into 1. This normalization is applied to each attribute meaning that all features are values roughly around 0. We do this normalization to aid in machine learning algorithms evaluating each feature equally. The functions used to apply this unit standard deviation normalization is also delivered as an output since the same normalization function needs to be applied to all data within the same machine learner. In the case of new data or data kept separate for testing purposes this normalization function produced as output is used instead of calculating one for just the new data.

3.8 Step 7. Machine Learner

With all the pre-processing steps finished we now have data in a machine learner ready format. Having the feature sets with their target answers in hand, the actual machine learning is fairly straightforward. The problem we are aiming to solve is the matching of similar columns. This is a classification task with each target being an attribute to which samples are associated to. We have a number of possible machine learning algorithms we can use for this task. Each of these algorithms also have some associated hyperparameters with a number of possible values. To determine the most optimal hyperparameter values we use grid search to exhaustively search for the best results within the training set.

Our possible machine learning algorithms with their associated hyperparameters are:

- KNearestNeighbour with N_neighbours for 1 to 31 for each odd number and metric with Euclidean and Minkowski Hyperparameters. This algorithm works by examining the closest neighbours for what attribute they belong to and based on that information predicts the attribute the feature set being evaluated belongs to. N_neighbours determines how many nearest neighbours should be checked. The metric parameter determines how nearest is calculated.

- LogisticRegression with a C of 10-3 through 103, increasing in steps of 1 exponent. This algorithm works by a series of boolean tests based on values of features to divide the dataset between the attributes.
- LinearSVC with a C of 10-3 through 103, increasing in steps of 1 exponent. This algorithm is a support vector classification (SVC) based algorithm, which means that attempts to divide the feature space between the different attributes by the use of function based lines. LinearSVC uses linear functions to define these lines.
- PolynominalSVC with a C of 10-3 through 103, increasing in steps of 1 exponent and a degree of 3.This algorithm is a support vector classification based algorithm, which means that attempts to divide the feature space between the different attributes by the use of function based lines. PolynominalSVC uses polynominal functions to define these lines.
- RbfSVC with a C of 10-3 through 103, increasing in steps of 1 exponent. This algorithm is a support vector classification based algorithm, which means that attempts to divide the feature space between the different attributes by the use of function based lines. RbfSVC uses radial basis functions (Rbf) to define these lines.
- SigmoidSVC with a C of 10-3 through 103, increasing in steps of 1 exponent.This algorithm is a support vector classification based algorithm, which means that attempts to divide the feature space between the different attributes by the use of function based lines. SigmoidSVC uses sigmoid functions to define these lines.
- Multinomial Naive Bayes which is a Naive Bayes based classifier suitable for multiple features.

3.9 Summary

By following the steps described in this chapter we can take data from different data sources and make them matchable on a data similarity level by use of a classification based machine learner and data characteristic based feature sets which have been calculated from aggregated samples.

The steps can be summarized as such:

1. Translate the data sources into a single homogenous format.
2. Clean the data from garbage data
3. Determining what type of data each column is
4. Determine the numerical values for numerical data
5. Dividing the data into samples

6. Calculating features over those samples

7. Use the feature sets in a classification based machine learner

For a number of these steps we have a number of possible techniques and we will evaluate which of these techniques perform best in section 4

Chapter 4

Experimentation

The previous chapter presented a conceptual approach for schema matching on the basis of "shape" of the data. The approach has several steps that can be technically realized with different algorithms and techniques. In this chapter, we experimentally evaluate these realization options to be able to choose the one(s) that works best. Secondly once we have made some determinations about which options to use we evaluate the full methodology on the basis of our experimental results. As we are working with our data set it is important to describe it in more detail. Our data set has been collected by a webscraper which has scraped a number of webshops that sell ball bearings. In total our data set consists of data from 9 different data sources. While most of the harvested data does concern ball bearings not all of them do but we have made efforts in the form of filtering to restrict the data to just bearings. Each of our data sources is structured as a JSON file with each product being a separate entry, each containing their respective properties as child entries within each product entry. In total our data set contains roughly 18500 products each of which have a number of properties for roughly 225000 data values. When considering properties overall we have 75 different properties though the majority of the data set is distributed over 33 different properties. In general we evaluate the success of our experiments on the basis of the score() function provided by the machine learner. This score is based on the ratio by which feature sets are evaluated to the correct class of properties and is defined as follows[1]:

n_{samples}

If $\hat{y}_i$ is the predicted value of the i-th sample and y_i is the corresponding true value, then the fraction of correct predictions over n_{samples} is defined as

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

[1] 3.3. Model evaluation: quantifying the quality of predictions scikit-learn 0.21.2 documentation, `https://scikit-learn.org/stable/modules/mode_evaluation.html#accuracy-score`, Accessed: 2019-06-18

where $1(x)$ is the indicator function.
To reduce the effect of any randomness within our experiments we execute each experiment 5 times and take a mean of these 5 scores. We also calculate a standard deviation over these 5 scores to be able to detect if we indeed have large amounts of deviation. In addition to the score and standard deviation we also collect a number of metadata values for the purposes of gaining an insight into the data we are working with at that in that exact experiment. We collect 6 different metadata values and collect them separately for textual and numerical data.

- The number of Samples
  - The exact amount different samples or feature sets that have been used as input. Using this value we can observe how many feature sets we have and is useful to understand the amount of data being used.
- Attribute Count
  - How many different properties each feature set can potentially be classified as. This is again useful for understanding the amount of data used as well as its distribution among different properties.
- Most Common Attribute
  - The property which has the largest representation within the data set. In general this is interesting comparatively.
- Most Common Attribute Count in samples
  - A more interesting measurement that the most common attribute is exactly how common it is. The biggest reason for this is the ZeroR score described below.
- ZeroR Score
  - A ZeroR Score is how accurate a classification algorithm would score when simply always classifying a sample or feature set as the most common class (Thus the Most Common Attribute). In general this is not that great of a classification algorithm but it provides a valuable baseline which can be used for comparison with how other algorithms score. A ZeroR Score is simple to calculate by dividing the Most Common Attribute Count with The number of Samples.
- Pure Random Score
  - The Pure Random Score is also a simple classification algorithm which can be used as a baseline comparison. The pure random algorithm works by simply randomly assigning each sample or feature to a available class. This means that the score of this algorithm is

calculated by dividing 1 by the number of available classes (Attribute Count).

We have a number of different experiments which can be divided into three broad categories.

- Sample Methods
  - One of the major ways in how our preprocessing can differ is in how it divides the data up into samples. In section 4.1 we extensively experiment to find the most suitable sampling method.
- Machine Learning Algorithms
  - Machine Learner Based Classification be be done using a large number of different algorithms. In section 4.2 we experiment to find the most suitable Machine Learning Algorithms.
- Prediction
  - In section 4.3 we shift from our arbitrary train-test split in our earlier experiments to a more realistic case of a split on a data source level and utilize the predict function of machine learning to evaluate a number of different splits.
  - Macro Prediction is a special subset of our prediction experiments where instead of scoring on a per sample level we scale it up one step and instead score on a column level. We do this by executing our prediction experiment normally but instead of taking the per sample score we instead analyze if the majority of samples are correctly predicted on a per column basis.

4.1 Experiments in varying Sampling Methods

One important aspect of our preprocessing is the creation of samples. By creating these samples we ensure that there is enough data to work with. It is important to ensure that resampling is not overused, i.e., does not sample the same values too many times. To test and validate the performance of our various potential sampling we performed experiments for each in turn. We kept our options on other levels the same. This means we always used all the data sources, used the nearest neighbour algorithm. Initially we set up experiments to test the data split, resampling without reuse within the same sample, and resampling with reuse within the same sample. A range of sample sizes was tested. For both resampling methodologies we tested varying numbers of samples to generate.

4.1.1 Full Dataset with randomized samples divided into fixed size groups

Hypothesis: We expect that different sample sizes should not affect the performance of the methodology, however a too small sample size would not give an accurate idea of the data and should thus perform poorly. Similarly a too large sample size may provide too few data points.

Experiment Setup: We process the full Deep Grooved Ball Bearing dataset using our preprocessing steps with the sampleizer set up to split the data into chunks of a fixed size discarding any remnants and generating all features. We transform those features to Unit Standard Deviation. We train and test the resultant processed data using a 90%/10% training/test split on a machine learner using the nearest neighbour algorithm using default hyperparameters.

Variations in the Experiment: We test with sample sizes of 5, 50, 100, 250, and 500. Any Column which does not meet a minimum size equal to the sample size will not generate samples since such column are too small to generate even one sample.

Measurements: For assessing the results of this experiment we perform the experiment 5 times and utilize the score() function from the machine learner to calculate a mean score and a standard deviation for those scores. To gain an insight into the data we are working with we also collect a number of metadata values. The meta data collected are:

- The number of Samples
- Attribute Count
- Most Common Attribute
- Most Common Attribute Count in samples
- ZeroR Score
- Pure Random Score

Experiment Results:

Sample Size	5	50	100	250	500
Textual Score	0.90	0.96	0.97	0.96	0.94
Textual Score Standard Deviation	0.003	0.003	0.002	0.008	0.010
Numerical Score	0.87	0.89	0.88	0.93	0.90
Numerical Score Standard Deviation	0.001	0.003	0.012	0.010	0.015

Table 4.1: Results of experiment Full Dataset with randomized samples divided into fixed size groups

Sample Size	5	50	100	250	500
Textual Feature Sample Number	37081	3608	1780	663	306
Textual Attribute Count	54	33	33	27	25
Textual Most Common Attribute	Product ID	Product ID	Product ID	Product ID	Product ID
Textual Most Common Attribute Count	6387	628	309	113	53
Textual ZeroR Score	0.17	0.17	0.17	0.17	0.17
Textual Pure Random Score	0.02	0.03	0.03	0.04	0.04
Numerical Feature Sample Number	10618	1012	489	172	79
Numerical Attribute Count	21	10	9	6	6
Numerical Most Common Attribute	Width	Width	Width	Width	Width
Numerical Most Common Attribute Count	3206	311	150	54	25
Numerical ZeroR Score	0.30	0.31	0.31	0.31	0.32
Numerical Pure Random Score	0.05	0.1	0.11	0.17	0.17

Table 4.2: Metadata results of experiment Full Dataset with randomized samples divided into fixed size groups

Experiment Discussion: When looking at the results of the data we can see several things. Firstly sample size 5 and 500 do not perform as bad as expected but do still perform worse. both of these sample sizes have lower textual scores scores than sample sizes 50, 100, 250 though sample size 500 does have the second highest numerical score. The ZeroR score remains consistent across all sample sizes. There is a small amount of variation due to the sample size acting as a minimum size requirement and not all columns being able to meet this requirement. Between sample size 5 and 50 there is a significant decrease in the number of attributes. This indicates that those attributes do not have any data columns associated with them of a size larger than 50 values. This also means that those attributes are scarcely present. Looking at performance as well as the number of attributes present 50, 100 and 250 as sample sizes perform best. The most common attribute remains the same across all sample sizes which is logical. It seems that roughly 1/6th of the textual data values is product ID and roughly 1/3rd of the numerical data is width data. A potential issue when using this methodology however is that we have relatively few samples to work with.

4.1.2 Full Dataset with fixed size samples created from resampling the data

Hypothesis: We expect that different sample sizes should not affect the performance of the methodology,however a too small sample size would not give an

accurate idea of the data and should thus perform poorly. Similarly a too large sample size may provide too few data points.

Experiment Setup: We will process the full Deep Grooved Ball Bearing dataset using our preprocessing steps with the sampleizer set up to generate a fixed number of samples from each data source of fixed size and generating all features. The way each sample is generated is by collecting a random selection of data values equal to the sample size before returning all the collected values to the available pool. This means that within the same sample the same exact entry cannot be repeated but across all samples it may be used several times. We transform the generated features to Unit Standard Deviation. We will train and test the resultant processed data using a 90%/10% training/test split on a machine learner using the nearest neighbour algorithm using default hyperparameters.

Variations in the Experiment: We will test with 100 and 1000 samples being generated For each of those 2 sample counts we will generate samples sizes of 5, 50, 100, 250. and 500. Any Column which does not meet a minimum size equal to the sample size will not generate samples since such column are too small to generate even one sample.

Measurements: For assessing the results of this experiment we will perform the experiment 5 times and utilize the score() function from the machine learner to calculate a mean score and a standard deviation for those scores. To gain an insight into the data we are working with we also collect a number of metadata values. The meta data collected are:

- The number of Samples
- Attribute Count
- Most Common Attribute
- Most Common Attribute Count in samples
- ZeroR Score
- Pure Random Score

Experiment Results:

Sample Size	5	50	100	250	500
Textual Score	0.83	0.96	0.98	0.993	0.99
Textual Score Standard Deviation	0.006	0.006	0.002	0.001	0.001
Numerical Score	0.79	0.92	0.92	0.91	0.90
Numerical Score Standard Deviation	0.002	0.004	0.003	0.003	0.004

Table 4.3: Results of experiment Full Dataset with fixed size samples created from resampling the data for 100 samples.

Sample Size	5	50	100	250	500
Textual Feature					
Sample Number	12000	8200	7800	5800	4800
Textual Attribute Count	54	33	33	27	25
Textual Most Common Attribute	Product ID	Product ID	Product ID	Product ID	Product ID
Textual Most Common Attribute Count	1600	1400	1300	900	700
Textual ZeroR Score	0.13	0.17	0.17	0.16	0.15
Textual Pure Random Score	0.019	0.03	0.03	0.04	0.04
Numerical Feature Sample Number	5400	3100	2600	1600	1300
Numerical Attribute Count	21	10	9	6	6
Numerical Most Common Attribute	Width	Width	Width	Width	Width
Numerical Most Common Attribute Count	1200	900	700	500	400
Numerical ZeroR Score	0.22	0.29	0.27	0.31	0.31
Numerical Pure Random Score	0.05	0.1	0.11	0.17	0.17

Table 4.4: Metadata results of experiment Full Dataset with fixed size samples created from resampling the data for 100 samples.

Sample Size	5	50	100	250	500
Textual Score	0.85	0.97	0.98	0.99	0.99
Textual Score Standard Deviation	0.006	0.003	0.001	0.002	0.002
Numerical Score	0.81	0.94	0.9	0.92	0.90
Numerical Score Standard Deviation	0.002	0.001	0.002	0.002	0.002

Table 4.5: Results of experiment Full Dataset with fixed size samples created from resampling the data for 1000 samples.

Sample Size	5	50	100	250	500
Textual Feature Sample Number	120000	82000	78000	58000	48000
Textual Attribute Count	54	33	33	27	25
Textual Most Common Attribute	Product ID	Product ID	Product ID	Product ID	Product ID
Textual Most Common Attribute Count	16000	14000	13000	9000	7000
Textual ZeroR Score	0.13	0.17	0.17	0.16	0.15
Textual Pure Random Score	0.02	0.03	0.03	0.04	0.04
Numerical Feature Sample Number	54000	31000	26000	16000	13000
Numerical Attribute Count	21	10	9	6	6
Numerical Most Common Attribute	Width	Width	Width	Width	Width
Numerical Most Common Attribute Count	12000	9000	7000	5000	4000
Numerical ZeroR Score	0.22	0.29	0.27	0.312	0.31
Numerical Pure Random Score	0.05	0.1	0.11	0.17	0.17

Table 4.6: Metadata results of experiment Full Dataset with fixed size samples created from resampling the data for 1000 samples.

Experiment Discussion: While the final average scores of this method are higher than the earlier data split method we can also note that as the number of samples increase we are having a convergent effect. Examining the ZeroR and the ratio between the number of samples and the most common attribute count shows that this is changing significantly as the sample size increases. Comparing this to the data split method where this ratio (and thus the ZeroR score) was fairly consistent and we did not have as much of a convergent effect it seems likely that we have some distortion due to oversampling. Comparing the generation of 100 and 1000 samples is less significant due to this distortion but overall it seems that generating 1000 samples performs slightly better.

4.1.3 Full Dataset with fixed size samples created from resampling the data with repeat in the same data

Hypothesis: We expect that different sample sizes should not affect the performance of the methodology,however a too small sample size would not give an accurate idea of the data and should thus perform poorly. Similarly a too large sample size may provide too few data points. This specific form of sampling may additionally display quirks in datasets of insufficient size since it is now able to generate such data but the repetition may be too much.

Experiment Setup: We will process the full Deep Grooved Ball Bearing dataset using our preprocessing steps with the sampleizer set up to generate a fixed number of samples from each data source of fixed size and generating all

features. The way each sample is generated is by collecting a random selection of data values equal to the sample size with each data value being available for reuse within the same sample. This means that within the same sample the same exact entry can be repeated as well as across all samples. We transform the generated features to Unit Standard Deviation. We will train and test the resultant processed data using a 90%/10% training/test split on a machine learner using the nearest neighbour algorithm using default hyperparameters.

Variations in the Experiment: We will test with 100 and 1000 samples being generated For each of those 2 sample counts we will generate samples sizes of 5, 50, 100, 250, and 500. Any Column which does not meet a minimum size equal to the sample size will not generate samples since such column are too small to generate even one sample.

Measurements: For assessing the results of this experiment we will perform the experiment 5 times and utilize the score() function from the machine learner to calculate a mean score and a standard deviation for those scores. To gain an insight into the data we are working with we also collect a number of metadata values. The meta data collected are:

- The number of Samples
- Attribute Count
- Most Common Attribute
- Most Common Attribute Count in samples
- ZeroR Score
- Pure Random Score

Experiment Results:

Sample Size	5	50	100	250	500
Textual Score	0.82	0.96	0.97	0.99	0.99
Textual Score Standard Deviation	0.003	0.004	0.005	0.001	0.002
Numerical Score	0.78	0.92	0.92	0.91	0.89
Numerical Score Standard Deviation	0.002	0.004	0.007	0.003	0.003

Table 4.7: Results of experiment Full Dataset with fixed size samples created from resampling the data with repeat in the same data for 100 samples.

Sample Size	5	50	100	250	500
Textual Feature Sample Number	12000	8200	7800	5800	4800
Textual Attribute Count	54	33	33	27	25
Textual Most Common Attribute	Product ID	Product ID	Product ID	Product ID	Product ID
Textual Most Common Attribute Count	1600	1400	1300	900	700
Textual ZeroR Score	0.13	0.17	0.17	0.16	0.15
Textual Pure Random Score	0.02	0.03	0.03	0.04	0.04
Numerical Feature Sample Number	5400	3100	2600	1600	1300
Numerical Attribute Count	21	10	9	6	6
Numerical Most Common Attribute	Width	Width	Width	Width	Width
Numerical Most Common Attribute Count	1200	900	700	500	400
Numerical ZeroR Score	0.22	0.29	0.27	0.31	0.31
Numerical Pure Random Score	0.05	0.10	0.11	0.17	0.17

Table 4.8: Metadata results of experiment Full Dataset with fixed size samples created from resampling the data with repeat in the same data for 100 samples.

Sample Size	5	50	100	250	500
Textual Score	0.84	0.97	0.98	0.99	0.99
Textual Score Standard Deviation	0.007	0.003	0.001	0.002	0.000
Numerical Score	0.81	0.94	0.94	0.92	0.90
Numerical Score Standard Deviation	0.002	0.001	0.001	0.001	0.002

Table 4.9: Results of experiment Full Dataset with fixed size samples created from resampling the data with repeat in the same data for 1000 samples.

Sample Size	5	50	100	250	500
Textual Feature Sample Number	120000	82000	78000	58000	48000
Textual Attribute Count	54	33	33	27	25
Textual Most Common Attribute	Product ID	Product ID	Product ID	Product ID	Product ID
Textual Most Common Attribute Count	16000	14000	13000	9000	7000
Textual ZeroR Score	0.13	0.17	0.17	0.16	0.15
Textual Pure Random Score	0.02	0.03	0.03	0.03	0.04
Numerical Feature Sample Number	54000	31000	26000	16000	13000
Numerical Attribute Count	21	10	9	6	6
Numerical Most Common Attribute	Width	Width	Width	Width	Width
Numerical Most Common Attribute Count	12000	9000	7000	5000	4000
Numerical ZeroR Score	0.22	0.29	0.27	0.31	0.31
Numerical Pure Random Score	0.05	0.1	0.11	.0.17	0.17

Table 4.10: Metadata results of experiment Full Dataset with fixed size samples created from resampling the data with repeat in the same data for 1000 samples.

Experiment Discussion: Our observations of this experiment shows mostly the same result as the resampling method without reuse and the discussion for that experiment also applies to this experiment.

On the basis of these experiments we see that the resampling methods have issues and are not suitable in their current state. The data split sampling method does seem to work as a base line method but 1700 and 500 samples for textual and numerical data respectively seems like a small number.. We have a number of issues which we need to address if we want to create an improved sampling method capable of generating more samples than a data split method. We cant simply resample each data column the same amount of time since if we do so we lose the proportionality of the data and overtrain on too similar samples causing a convergent effect. Finally we noted that we do not establish a lower bound on the data split method resulting in some attributes having only a few samples associated with them. This number of samples is so low that a machine learner cannot be effectively trained on those samples but the attribute is still present in the machine learner. Having determined these issues we devised an additional sampling method to alleviate these issues. This method is based on our resampling without reuse within the same sample method however we have set additional requirements on the minimum size of the dataset as well as an additional limiter on the number of samples that can be created based on the size of the dataset.

4.1.4 Issues found based on the tested sampling methods

Found Issues: Based on the previous experiments we have found several potential issues with our current methods of sampling. The current requirements we have set for the data to be considered suitable for use may be incorrect. This manifests itself into a variety of issues:

- When generating samples for the split into fixed group methodology small data samples may generate too few samples meaning that there are is not enough representation of some attributes when creating the training splits.
- In the various resampling methods smaller datasets (not much bigger than the sample size) may result in too much repeat in the sampled data since from that small dataset all generated samples are nearly identical, meaning that scores are artificially inflated and that there is a measure of overtraining.
- This issue with smaller datasets cuts both ways since we are generating the same number of samples regardless of how large each dataset is. Since we are working with multiple datasets each of the same attribute this means that proportionality is lost. While the effect of this is difficult to determine we do feel that maintaining this on some level is important.

Having determined these issues we devised an additional sampling method to alleviate these issues. This method is based on our second sampling method however we have set additional requirements on the minimum size of the dataset as well as an additional limiter on the number of samples that can be created based on the size of the dataset.

4.1.5 Issues found based on the tested sampling methods

Found Issues: Based on the previous experiments we have found several potential issues with our current methods of sampling. The current requirements we have set for the data to be considered suitable for use may be incorrect. This manifests itself into a variety of issues:

- When generating samples for the split into fixed group methodology small data samples may generate too few samples meaning that there are is not enough representation of some attributes when creating the training splits.
- In the various resampling methods smaller datasets (not much bigger than the sample size) may result in too much repeat in the sampled data since from that small dataset all generated samples are nearly identical, meaning that scores are artificially inflated and that there is a measure of overtraining.
- This issue with smaller datasets cuts both ways since we are generating the same number of samples regardless of how large each dataset is. Since we are working with multiple datasets each of the same attribute this

means that proportionality is lost. As we are not trying to compensate for distribution of data and even of that case this is a rather unguided method we want to maintain this proportionality.

Having determined these issues we devised an additional sampling method to alleviate these issues. This method is based on our second sampling method however we have set additional requirements on the minimum size of the dataset as well as an additional limiter on the number of samples that can be created based on the size of the dataset.

4.1.6 Full Dataset with fixed size samples created from resampling the data with additional limiters

When using our new revised method we once again utilize sampling with returning the data to the pool once each feature is generated however we have added two additional limiters. For a group of data to be allowed to be sampled it must be at least 2 times the sample size (50 sample size means a minimum dataset size of 100) and to attempt to keep some proportionality a group of data may only be used to generate samples equal to 3 times the dataset size divided by the sample size (in our example of a sample size of 50 with a data group size of 100 this mean 6 samples of 50 each (3*100/50=6)).

Hypothesis: We believe that with these additional limiters the revised sampling method will score worse than the previous resampling methods but that this will mainly be due to the fact that the previous resampling methods were working with distorted sample data resulting in artificially higher scores. This revised method should perform roughly similar to the data split method but should be more robust than that methodology due to the resampling.

Experiment Setup: We will process the full Deep Grooved Ball Bearing dataset using our preprocessing steps with the sampleizer set up to generate a fixed number of samples from each data source of fixed size and generating all features. The way each sample is generated is by collecting a random selection of data values equal to the sample size before returning all the collected values to the available pool. This means that within the same sample the same exact entry cannot be repeated but across all samples it may be used several times. We transform the generated features to Unit Standard Deviation. We will train and test the resultant processed data using a 3 fold cross validation on a machine learner using the nearest neighbour algorithm with grid searching hyperparameters for each uneven amount of neighbours from 1 to 31 and using .euclidean or minkowski distance metrics.

Variations in the Experiment: We will test with 100 and 1000 samples being generated. For each of those sample numbers we will generate samples sizes of 5, 50, 100, 250, and 500.

Measurements: For assessing the results of this experiment we will perform the experiment 5 times and utilize the score() function from the machine learner to calculate a mean score and a standard deviation for those scores. To gain an

insight into the data we are working with we also collect a number of metadata values. The meta data collected are:

- The number of Samples
- Attribute Count
- Most Common Attribute
- Most Common Attribute Count in samples
- ZeroR Score
- Pure Random Score

Experiment Results:

Sample Size	5	50	100	250	500
Textual Score	0.83	0.95	0.97	0.99	0.99
Textual Score Standard Deviation	0.011	0.005	0.006	0.004	0.007
Numerical Score	0.77	0.83	0.86	0.91	0.90
Numerical Score Standard Deviation	.0071	0.017	0.014	0.020	0.032

Table 4.11: Results of experiment Full Dataset with fixed size samples created from resampling the data with additional limiters 100 samples.

Sample Size	5	50	100	250	500
Textual Feature Sample Number	10044	5956	4523	2033	956
Textual Attribute Count	54	33	33	27	25
Textual Most Common Attribute	Product ID	Product ID	Product ID	Product ID	Product ID
Textual Most Common Attribute Count	1524	962	709	343	161
Textual ZeroR Score	0.15	0.16	0.16	0.17	0.17
Textual Pure Random Score	0.019	0.03	0.03	0.05	0.04
Numerical Feature Sample Number	4279	1806	1241	532	247
Numerical Attribute Count	21	10	9	6	6
Numerical Most Common Attribute	Width	Width	Width	Width	Width
Numerical Most Common Attribute Count	1110	535	372	168	78
Numerical ZeroR Score	0.26	0.30	0.30	0.32	0.32
Numerical Pure Random Score	0.05	0.10	0.11	0.17	0.17

Table 4.12: Metadata results of experiment Full Dataset with fixed size samples created from resampling the data with additional limiters for 100 samples.

Sample Size	5	50	100	250	500
Textual Score	0.88	0.97	0.97	0.99	0.99
Textual Score Standard Deviation	0.005	0.003	0.004	0.005	0.004
Numerical Score	0.80	0.88	0.88	0.91	0.91
Numerical Score Standard Deviation	0.004	0.011	0.011	0.022	0.031

Table 4.13: Results of experiment Full Dataset with fixed size samples created from resampling the data with additional limiters for 1000 samples.

Sample Size	5	50	100	250	500
Textual Feature Sample Number	61525	10917	5413	2033	956
Textual Attribute Count	54	33	33	27	25
Textual Most Common Attribute	Product ID	Product ID	Product ID	Product ID	Product ID
Textual Most Common Attribute Count	9763	1900	939	343	161
Textual ZeroR Score	0.16	0.17	0.17	0.17	0.17
Textual Pure Random Score	0.02	0.03	0.03	0.04	0.04
Numerical Feature Sample Number	19309	3058	1484	532	247
Numerical Attribute Count	21	10	9	6	6
Numerical Most Common Attribute	Width	Width	Width	Width	Width
Numerical Most Common Attribute Count	5580	938	453	168	78
Numerical ZeroR Score	0.29	0.31	0.31	0.32	0.32
Numerical Pure Random Score	0.05	0.10	0.11	0.17	0.17

Table 4.14: Metadata results of experiment Full Dataset with fixed size samples created from resampling the data with additional minimals and maximals for 1000 samples.

Experiment Discussion: The methodology of resampling with limiters again scores well but comparing to our other methodologies it actually scores the worst with the exception of data split when considering higher sample sizes. Looking at the metadata however this seems to be our most promising method so far. Firstly unlike the previous resampling methods the distortion we noticed in the previous resampling methods is significantly less present here. The ZeroR score (which originates from the ratio between the number of samples and the most common attribute) is far more consistent with the results from the data split method. The main issue we had with using the data split method is also alleviated in that for the samples sizes of 100 and upwards roughly 3 times as many samples are generated. This combined with the fact that when increasing the target of generated samples per column from 100 to 1000 there is only a 20% increase in the number of generated samples in the case of sample size 100 and no increase for 250 and 500 sample sizes indicates to us that for most the main limiting factor in the number of samples is the new limiter we imposed on how many samples can be generated from a column based on the ratio between the number of data values within the column and the sample size which with our setting translated to 3 times that number.

4.1.7 Graphical Comparison between sampling methods

To improve the readability and aid in the comparison of the various methods we have generated a number of graphs from the tabulated scores of each sampling method.

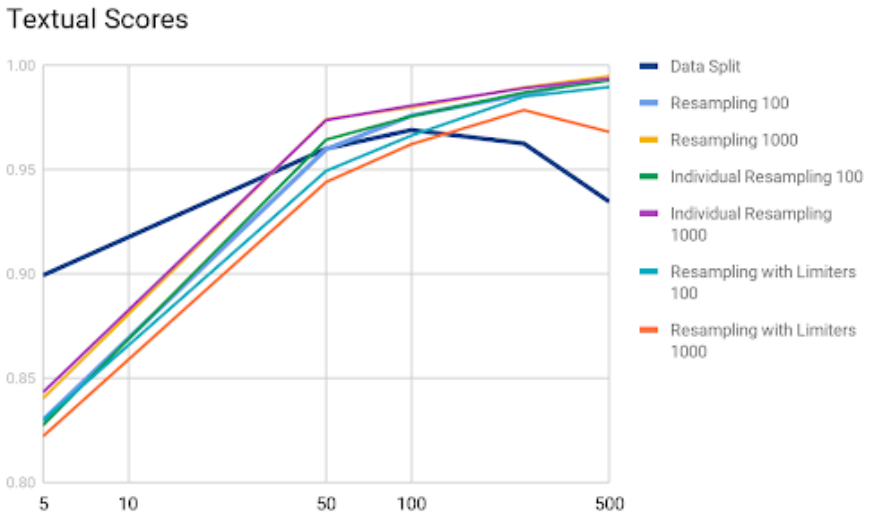

Figure 4.1: Graph of the textual scores for all previous sampling experiments.

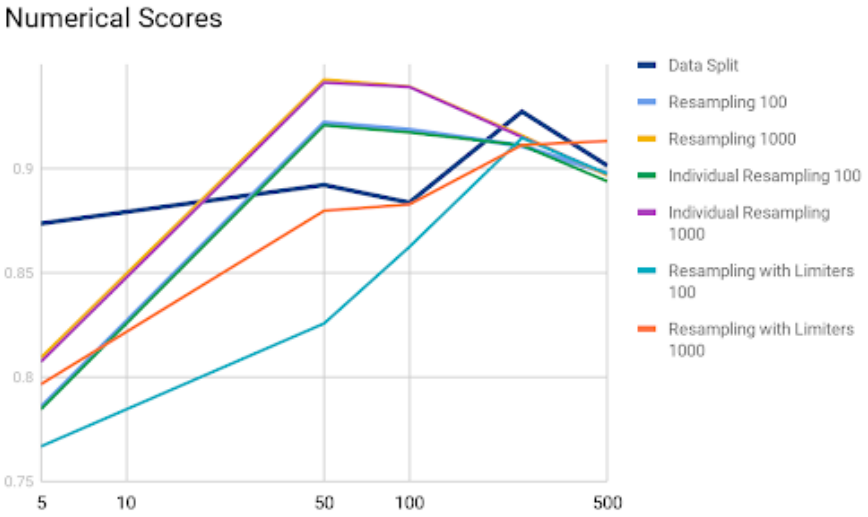

Figure 4.2: Graph of the textual scores for all previous sampling experiments.

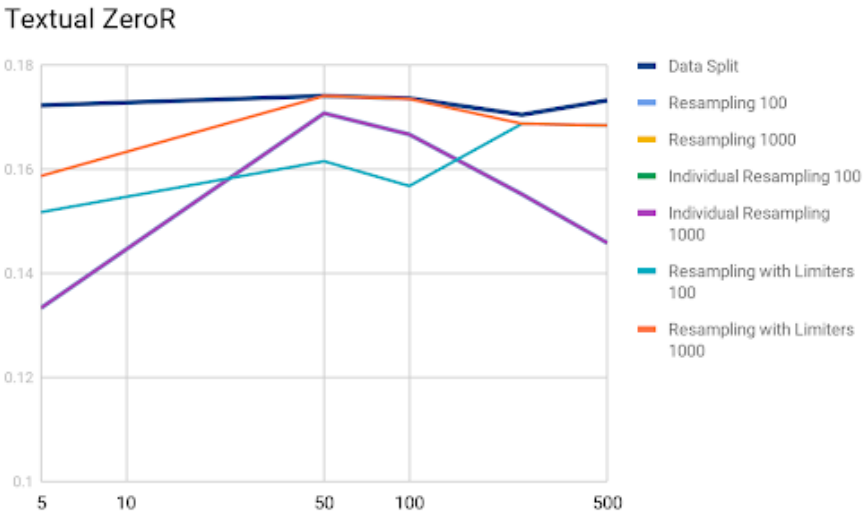

Figure 4.3: Graph of the ZeroR textual scores for all previous sampling experiments. Resampling 100, Resampling 1000, Individual Resampling 100 and Individual Resampling 1000 are all identical in this graph.

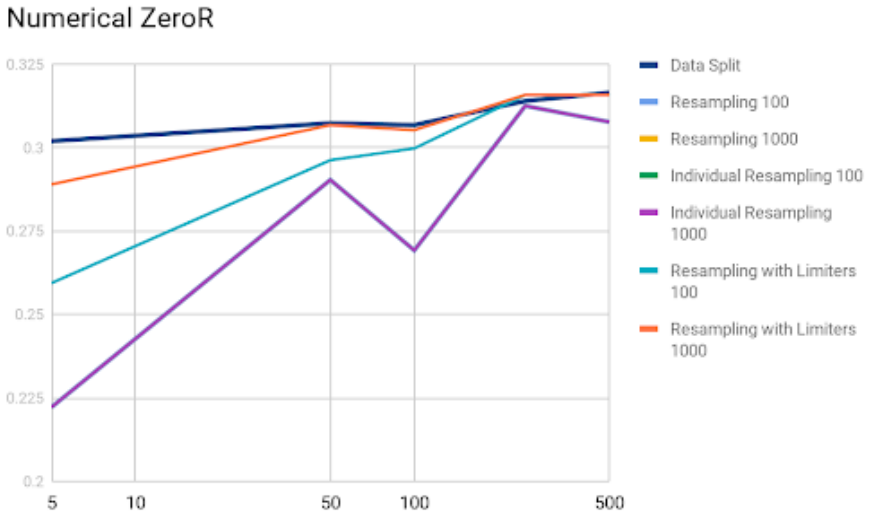

Figure 4.4: Graph of the ZeroR textual scores for all previous sampling experiments. Resampling 100, Resampling 1000, Individual Resampling 100 and Individual Resampling 1000 are all identical in this graph.

As discussed in the individual experiment discussions we can draw a number conclusions on the basis of these graphs. Firstly as we initially hypothesized low sample sizes perform poorly but to our surprise even the sample size of 50 performs fairly well though the larger sample sizes do perform better. When considering larger sample sizes the various methodologies all perform with higher scores with the exception of the data split and resampling with limiters methods. This also brings us to an issue with the resampling methods where they seem to be overtraining or becoming distorted in different ways. This conclusion is also supported when examining the ZeroR graphs which show that with the exception of the resampling with limiters with 1000 samples to be generated there is a significant deviance from the data split method with can be considered the default. This suggests to use one of those two methodologies considering that the other methodologies are undesirably distorting the data. Comparing between these two methodologies suggests to use the resampling with limiters methodology since this increases the number of available samples which generally desirable with machine learning. A final observation is that with the current testing method and machine learning algorithm matching textual data seems easier than matching numerical data. This is unlike our expectations but we will not draw any conclusions from this at this point since further experiments will provide more information.

From examining these graphs and the associated tables we can draw a number of conclusions.

- As hypothesized low samples sizes perform poorly but even a sample size of 50 performs fairly well.
- Larger samples size perform poorly when looking at the data split method but in the resampling methods this only leads to an increased performance, if in our new resampling method with limiters to a lesser extent.
- Textual data seems easier to match than Numerical Data.
- There are fairly large ZeroR deviations in the resampling methods compared to the data split method with the exception of the resampling with limiters method with a sample number of a 1000. To us this seems like a good indicator of the distortion discussed earlier.

Based upon these results using our resampling with limiters method with a sample size of 250 and a sample number of 1000 seems to have the best results.

4.1.8 Comparison with the exact same datasets

Until this point we had one additional factor of difference we had not counted on. For each of these sample sizes and sample counts we have been working with different restrictions on the data due to requirements of each algorithm. Thus to truly test and see what algorithm performs best we should test the algorithms again but with a set of data that satisfies all requirement. To achieve this we set up a requirement of a minimum size of 750 for each dataset to be considered and tested each algorithm with this data for the sample sizes of 50,100 and 250.

Hypothesis: We still expect roughly the same results as our previous experiments showed but this experiment should provide important confirmation.

Experiment Setup: We will process the full Deep Grooved Ball Bearing dataset with a minimum size requirement to columns of 750 data values using our preprocessing steps with the sampleizer set up to generate samples based on our various sampling algorithms and generating all features. We transform the generated features to Unit Standard Deviation. We will train and test the resultant processed data using a 3 fold cross validation on a machine learner using the nearest neighbour algorithm with grid searching hyperparameters for each uneven amount of neighbours from 1 to 31 and using .euclidean or minkowski distance metrics.

Variations in the Experiment: We will generate sample sizes of 50, 100 and 250. We will use the Data Split, Resampling, Individual Resampling and Resamping with Limiters as our sampling techniques. We will be generating 100 and 1000 samples per column in the case of our resampling algorithms

Measurements: For assessing the results of this experiment we will perform each experiment 5 times and utilize the score() function from the machine learner to calculate a mean score and a standard deviation for those 5 scores. To gain an insight into the data we are working with we also collect a number of metadata values. The meta data collected are:

- The number of Samples

- Attribute Count
- Most Common Attribute
- Most Common Attribute Count in samples
- ZeroR Score
- Pure Random Score

Experiment Results:

Sample Size	50	100	250
Textual Feature Number of Samples	3406	1691	663
Textual Attribute Count	27	27	27
Textual Most Common Attribute	Product ID	Product ID	Product ID
Textual Most Common Attribute Count	577	286	113
Textual ZeroR Score	0.17	0.17	0.17
Textual Pure Random Score	0.04	0.04	0.04
Numerical Feature Sample Number	893	443	172
Numerical Attribute Count	6	6	6
Numerical Most Common Attribute	Width	Width	Width
Numerical Most Common Attribute Count	282	140	54
Numerical ZeroR Score	0.32	0.32	0.32
Numerical Pure Random Score	0.17	0.17	0.17

Table 4.15: Metadata results for the Data Split technique when filtering for a minimum column size of 750 data records.

Sample Size	50	100	250
Textual Score	0.96	0.97	0.97
Textual Score Standard Deviation	0.005	0.007	0.015
Numerical Score	0.89	0.89	0.88
Numerical Score Standard Deviation	0.011	0.021	0.042

Table 4.16: Results for the Data Split technique when filtering for a minimum column size of 750 data records.

Sample Size	50	100	250
Textual Feature Sample Number	5800	5800	5800
Textual Attribute Count	27	27	27
Textual Most Common Attribute	Product ID	Product ID	Product ID
Textual Most Common Attribute Count	900	900	900
Textual ZeroR Score	0.16	0.16	0.16
Textual Pure Random Score	0.04	0.04	0.04
Numerical Feature Sample Number	1600	1600	1600
Numerical Attribute Count	6	6	6
Numerical Most Common Attribute	Width	Width	Width
Numerical Most Common Attribute Count	500	500	500
Numerical ZeroR Score	0.31	0.31	0.31
Numerical Pure Random Score	0.17	0.17	0.17

Table 4.17: Metadata results for the technique algorithm with 100 samples when filtering for a minimum column size of 750 data records.

Sample Size	50	100	250
Textual Score	0.96	0.97	0.98
Textual Score Standard Deviation	0.006	0.004	0.003
Numerical Score	0.83	0.86	0.87
Numerical Score Standard Deviation	0.012	0.015	0.014

Table 4.18: Results for the Resampling technique with 100 samples when filtering for a minimum column size of 750 data records.

Sample Size	50	100	250
Textual Feature Sample Number	58000	58000	58000
Textual Attribute Count	27	27	27
Textual Most Common Attribute	Product ID	Product ID	Product ID
Textual Most Common Attribute Count	9000	9000	9000
Textual ZeroR Score	0.16	0.16	0.16
Textual Pure Random Score	0.004	0.04	0.04
Numerical Feature/Sample Number	16000	16000	16000
Numerical Attribute Count	6	6	6
Numerical Most Common Attribute	Width	Width	Width
Numerical Most Common Attribute Count	5000	5000	5000
Numerical ZeroR Score	0.31	0.31	0.31
Numerical Pure Random Score	0.17	0.17	0.17

Table 4.19: Metadata results for the Resampling technique with 1000 samples when filtering for a minimum column size of 750 data records.

Sample Size	50	100	250
Textual Score	0.97	0.98	0.99
Textual Score Standard Deviation	0.004	0.005	0.002
Numerical Score	0.85	0.87	0.87
Numerical Score Standard Deviation	0.005	0.003	0.005

Table 4.20: Results for the Resampling technique with 1000 samples when filtering for a minimum column size of 750 data records.

Sample Size	50	100	250
Textual Feature Sample Number	5800	5800	5800
Textual Attribute Count	27	27	27
Textual Most Common Attribute	Product ID	Product ID	Product ID
Textual Most Common Attribute Count	900	900	900
Textual ZeroR Score	0.16	0.16	0.16
Textual Pure Random Score	0.04	0.04	0.04
Numerical Feature Sample Number	1600	1600	1600
Numerical Attribute Count	6	6	6
Numerical Most Common Attribute	Width	Width	Width
Numerical Most Common Attribute Count	500	500	500
Numerical ZeroR Score	0.31	0.31	0.31
Numerical Pure Random Score	0.17	0.17	0.17

Table 4.21: Metadata results for the Individual Resampling technique with 100 samples when filtering for a minimum column size of 750 data records.

Sample Size	50	100	250
Textual Score	0.96	0.97	0.98
Textual Score Standard Deviation	0.005	0.004	0.002
Numerical Score	0.83	0.85	0.86
Numerical Score Standard Deviation	0.011	0.008	0.014

Table 4.22: Results for the Individual Resampling technique with 100 samples when filtering for a minimum column size of 750 data records.

Sample Size	50	100	250
Textual Feature/Sample Number	58000	58000	58000
Textual Attribute Count	27	27	27
Textual Most Common Attribute	Product ID	Product ID	Product ID
Textual Most Common Attribute Count	9000	9000	9000
Textual ZeroR Score	0.16	0.16	0.16
Textual Pure Random Score	0.04	0.04	0.04
Numerical Feature/Sample Number	16000	16000	16000
Numerical Attribute Count	6	6	6
Numerical Most Common Attribute	Width	Width	Width
Numerical Most Common Attribute Count	5000	5000	5000
Numerical ZeroR Score	0.31	0.31	0.31
Numerical Pure Random Score	0.17	0.17	0.17

Table 4.23: Metadata results for the Individual Resampling technique with 1000 samples when filtering for a minimum column size of 750 data records.

Sample Size	50	100	250
Textual Score	0.97	0.98	0.99
Textual Score Standard Deviation	0.004	0.001	0.002
Numerical Score	0.85	0.87	0.87
Numerical Score Standard Deviation	0.006	0.005	0.003

Table 4.24: Results for the Individual Resampling technique with 1000 samples when filtering for a minimum column size of 750 data records.

Sample Size	50	100	250
Textual Feature Sample Number	5328	4245	2033
Textual Attribute Count	27	27	27
Textual Most Common Attribute	Product ID	Product ID	Product ID
Textual Most Common Attribute Count	804	640	343
Textual ZeroR Score	0.15	0.15	0.17
Textual Pure Random Score	0.04	0.04	0.04
Numerical Feature Sample Number	1438	1098	532
Numerical Attribute Count	6	6	6
Numerical Most Common Attribute	Width	Width	Width
Numerical Most Common Attribute Count	446	342	168
Numerical ZeroR Score	0.32	0.32	0.32
Numerical Pure Random Score	0.17	0.17	0.17

Table 4.25: Metadata results for the Resampling with Limiters technique with 100 samples when filtering for a minimum column size of 750 data records.

Sample Size	50	100	250
Textual Score	0.96	0.97	0.98
Textual Score Standard Deviation	0.004	0.005	0.002
Numerical Score	0.82	0.89	0.91
Numerical Score Standard Deviation	0.012	0.016	0.011

Table 4.26: Results for the Resampling with Limiters technique with 100 samples when filtering for a minimum column size of 750 data records.

Sample Size	50	100	250
Textual Feature Sample Number	10289	5135	2033
Textual Attribute Count	27	27	27
Textual Most Common Attribute	Product ID	Product ID	Product ID
Textual Most Common Attribute Count	1742	870	343
Textual ZeroR Score	0.17	0.17	0.17
Textual Pure Random Score	0.04	0.04	0.04
Numerical Feature Sample Number	2690	1341	532
Numerical Attribute Count	6	6	6
Numerical Most Common Attribute	Width	Width	Width
Numerical Most Common Attribute Count	849	423	168
Numerical ZeroR Score	0.32	0.32	0.32
Numerical Pure Random Score	0.17	0.17	0.17

Table 4.27: Metadata results for the Resampling with Limiters technique with 1000 samples when filtering for a minimum column size of 750 data records.

Sample Size	50	100	250
Textual Score	0.97	0.98	0.99
Textual Score Standard Deviation	0.003	0.004	0.003
Numerical Score	0.90	0.91	0.91
Numerical Score Standard Deviation	0.009	0.014	0.021

Table 4.28: Results for the Resampling with Limiters technique with 1000 samples when filtering for a minimum column size of 750 data records.

Experiment Discussion:

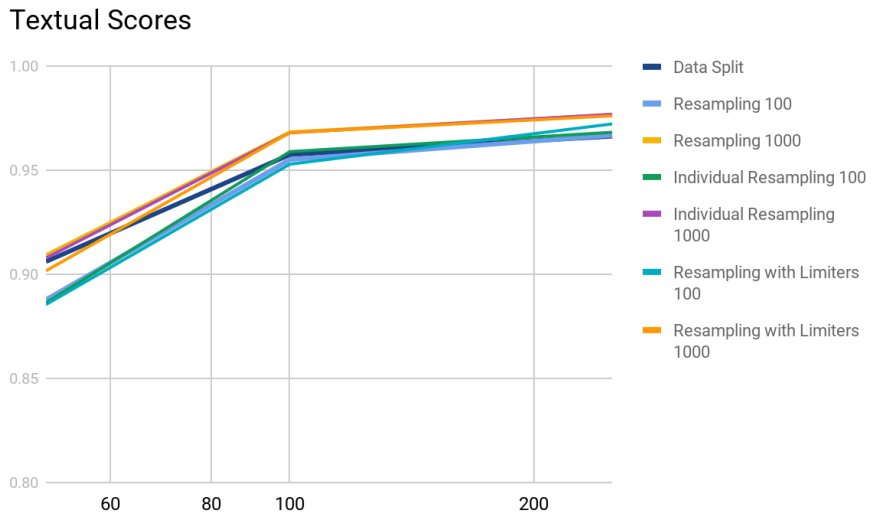

Figure 4.5: Graph of the textual scores for the sampling experiments with the same dataset by filtering for a minimum column size of 750 data records. Resampling 1000, Individual Resampling 1000 and Resampling with Limiters 1000 are all nearly identical in this graph.

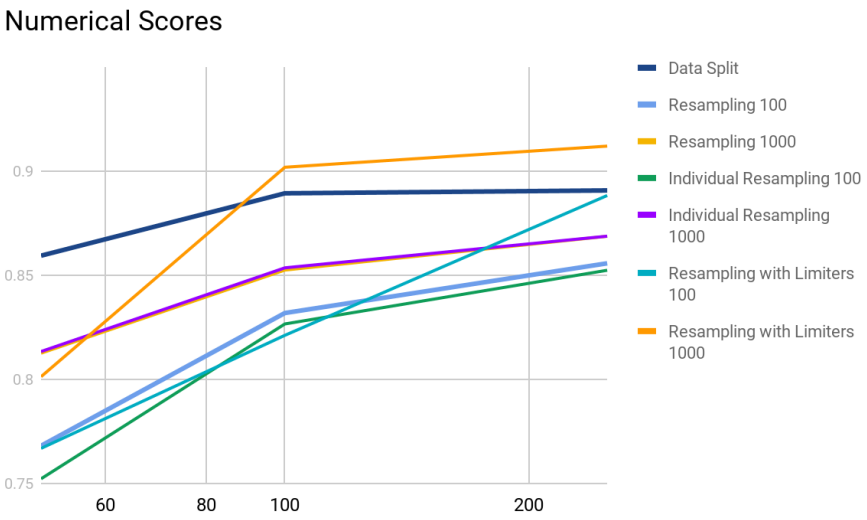

Figure 4.6: Graph of the textual scores for the sampling experiments with the same dataset by filtering for a minimum column size of 750 data records. Resampling 1000 and Individual Resampling 1000 are nearly identical in this graph.

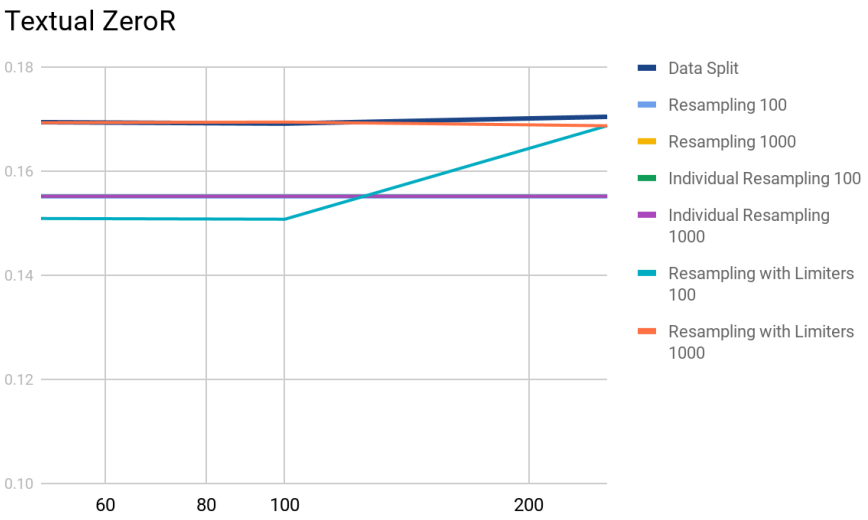

Figure 4.7: Graph of the ZeroR textual scores for the sampling experiments with the same dataset by filtering for a minimum column size of 750 data records. Resampling 1000, Individual Resampling 1000 and Resampling with Limiters 1000 are all nearly identical in this graph.

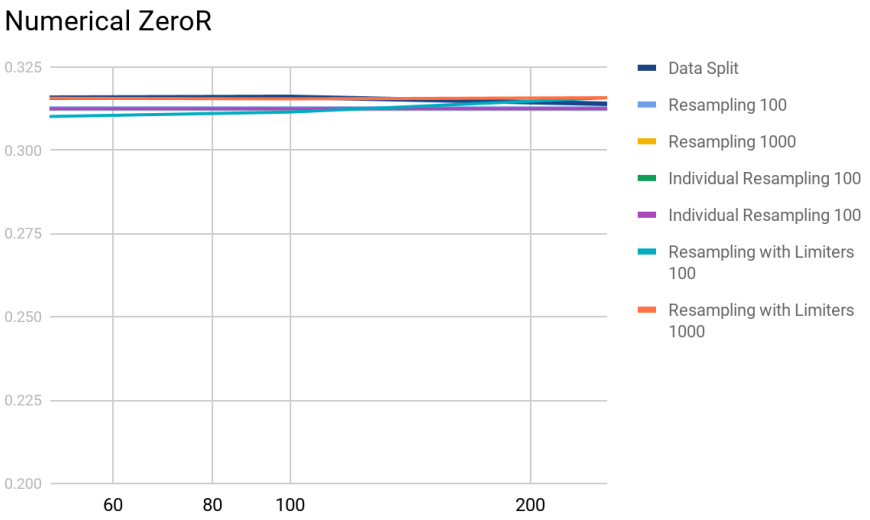

Figure 4.8: Graph of the ZeroR textual scores for the sampling experiments with the same dataset by filtering for a minimum column size of 750 data records. Resampling 1000, Individual Resampling 1000 and Resampling with Limiters 1000 are all identical in this graph.

From examining these graphs and the underlying experiment results we can again make some observations and draw conclusions.

- Looking at the textual ZeroR graph we can note that with the exception of Resampling with limiters with a 1000 samples to be generated all resampling methods stil deviate from the ZeroR from data split indicating that there is still some level of distortion happening.
- In the case of the numerical ZeroR all methods score roughly equal which is interesting.
  - A possible explanation for why the numerical ZeroR remains fairly consistent in this case unlike our other experiments where it seems to be a good indicator for the overall data being distorted can be found in the nature of our data and the fact that we have filtered our all columns smaller than 750 data values. The nature of our data is that of products which logically possess all applicable attributes to describe them. It could be that due to the extra filtering only those attributes which are core to the product remain. Meaning that each column is present roughly equally on a per data source level. Due to this it does not matter what sampling method we use in terms of the ratios of each column since the starting size is the same for each column.

- When considering the score graph for textual data all resampling methods which attempt to generate 1000 samples perform best indicating that a larger amount of samples to work with is beneficial for matching.
- Resampling with limiters with a 1000 samples to be generated has roughly the same amount of samples as the normal Resampling methods which generate 100 samples yet there are significant score differences.
- When considering the numerical score graph only Resampling with limiters with a 1000 samples to be generated scores better than the date split method.

From our observations so far its seem that Resampling with limiters with a 1000 samples to be generated is the best method. When considering sample sizes both 100 and 250 score similarly with a minor increase in score for the larger sample size with both being similar to the data split method when considering ZeroR. While the 250 sample size technically performed slightly better in these experiments we chose to use a sample size 100 on the basis that this will result in more samples to work with overall but still of sufficient size to characterize attributes.

From this we conclude that using the resampling method with limiters with a sample size of 100 and 1000 sample number performs the best. When comparing this to the unfiltered methodology this setup also seems to work there.

4.2 Experiments with using different machine learning algorithms

Thus far we have used the Nearest Neighbour algorithm for the matching in our machine learner but there are multiple classification algorithms available to us. To determine which algorithms perform best we performed experiments testing to determine which algorithms perform best.

Hypothesis: It is difficult for us to set expectations. We expect that KnearestNeighbour will perform well as it has been performing well so far and that situation should not change here. Other algorithms will probably also perform well but the differences between this are hard for use to predict at this point.

Experiment Setup: We will process the full Deep Grooved Ball Bearing dataset using our preprocessing steps with the sampleizer set up to generate up to a 1000 samples from each column using the Resampling with Limiters method. We will only consider columns which have a minimum size of 750 data values. We generate all features. We transform the generated features to Unit Standard Deviation. We will train and test the resultant processed data using a 3 fold cross validation on a machine learner using varying classification machine learning algorithms

Variations in the Experiment: We will test with the following machine learning algorithms.

- KNearestNeighbour with n_neighbours for 1 to 31 for each uneven number and metric euclidean and minkowski Hyperparameters
- LogisticRegression with a C of 10^{-3} through 10^{3}, increasing in steps of 1 exponent and using saga as solver.
- LinearSVC with a C of 10^{-3} through 10^{3}, increasing in steps of 1 exponent.
- PolynominalSVC with a C of 10^{-3} through 10^{3}, increasing in steps of 1 exponent and a degree of 3.
- RbfSVC with a C of 10^{-3} through 10^{3}, increasing in steps of 1 exponent.
- SigmoidSVC with a C of 10^{-3} through 10^{3}, increasing in steps of 1 exponent.
- Multinomial Naive Bayes

Measurements: For assessing the results of this experiment we will use the score() function from the machine learner.

Experiment Results:

Machine Learning Algorithm	KNearest Neighbour	Logistic Regression	Linear SVC	PolynominalRbf SVC SVC		Sigmoid SVC	Naive Bayes
Textual Score	0.98	0.90	0.88	0.97	0.98	0.58	0.48
Numerical Score	0.91	0.82	0.81	0.89	0.92	0.81	0.33

Table 4.29: Results of the Machine Learner Algorithm Experiments

Experiment Discussion: Examining our results we see that KNearest-Neighbour and a support vector machine using a radial basis function kernel perform best. The polynominal based support vector machine comes third losing some points in both textual and numerical cases. Of our remaining algorithms Sigmoid based support vector machines and Naive Bayes seem non viable but Logistic Regression and Linear based suppor vector machines seem workable.

On the basis of these results it is difficult to decide on a single algorithm due to the close results between KNearestNeighbour and Rbf SVC. We decided that we will test using both algorithms.

4.3 Experiments with prediction

So far we have not split our test and training data beforehand and simply used build in cross validation generators to split our data into a training set and a test set. This indicates to us that our theory that matching on the basis of data characteristics and commonality is correct on at least some level. Due to the way cross validation generators split data it is however possible that these

correct matches are made on the basis of similarity to samples from the same column. If this is the case this is already valuable and interesting to know but for more practical purposes we want to test the case where we split our test and training data on a data source level. This is the more practical case since the end application is the macro scale task of matching columns of the same property together. By splitting on a data source level we make it so that samples generated from the same column cannot be correctly classified on the basis of samples generated from that same column but instead on samples from other data sources which belong to the same property. In this new test we first train a classifier using the majority of our other data sources with a GridSearchCross-Validation before evaluating those separate testing sets according to the training data sources by using the predict function. We evaluate the performance on the basis of Metrics.accuracy_score.

Hypothesis: We may be generating self referencing positives due to samples generated from the same column being in both the training and the testing set. By performing experiments where the training and the testing set are composed of separate sources we should be able to determine if such self referencing positives are influencing our results as well as acquiring results for a more realistic test scenario where from an existing large data source with associated classifier an additional data source is added.

Experiment Setup: We split our full Deep Grooved Ball Bearing dataset into a training set and a test set. We will do this on a per data source level. We excluded the worse performing Motionindustries data source from the test set in the last two experiments to check if this would dramatically improve our matching scores. In the final experimental variation we filtered the test set in such a way that no columns with new attributes were included. We process each data set separately using our preprocessing steps. We create samples of 100 data value size and use the resampling with limiters method while attempting to generate 1000 samples for each column. In some cases we added a filtering to exclude columns that had less than 750 values. We will generate all features. We transform the generated features to Unit Standard Deviation. In the case of the test set we use the transformation function based on the training set. We use the processed training set data to train two machine learners using gridsearch. One machine learner uses the nearest neighbour algorithm with hyperparameters for each uneven amount of neighbours from 1 to 31 and using .euclidean or minkowski distance metrics. The other uses RbfSVC with a C of 10^{-3} through 10^{3}, increasing in steps of 1 exponent as its hyperparameter. Using the predict function from each of the classifiers we predict the test set.

Variations in the Experiment: We test a number of training and test set separations as well as a number of additional filters.

- Bearingsonline in the test set with all other data sources in the training set.
- Bearingsonline in the test set with all other data sources in the training set. Data filtered to minimum column size of 750.

- Bearingsdirect in the test set with all other data sources in the training set.
- Bearingsonline and Bearingsdirect in the test set with all other data sources in the training set.
- Bearingsonline and Bearingsdirect in the test set with all other data sources in the training set. Data filtered to minimum column size of 750.
- BTShop in the test set with all other data sources in the training set.
- Eriks and ABF in the test set with all other data sources in the training set.
- Xbearings in the test set with all other data sources in the training set. Data filtered to minimum column size of 750.
- Motionindustries in the test set with all other data sources in the training set.
- Xbearings and Bearingsdirect in the test set with all other data sources except motionindustries in the training set.
- Xbearings and Bearingsdirect in the test set with all other data sources except motionindustries in the training set. Data filtered such that only columns associated to attributes that are present in both test set and training set are used.

Measurements: For assessing the results of this experiment we will perform the experiment 5 times and utilize the metrics.accuracy_score function from the machine learner to determine the results of our predictions.To gain an insight into the data we are working with we also collect a number of metadata values. The meta data collected are:

- The number of Samples
- Attribute Count
- Most Common Attribute
- Most Common Attribute Count in samples
- ZeroR Score
- Pure Random Score

Experiment Results:

	Sample Number	Attribute Count	Most Common Attribute	Most Common Attribute Count	ZeroR	Pure Random
Training set Textual	5267	33	Product ID	915	0.17	0.03
Training set Numerical	1409	9	Width	428	0.30	0.11
Test set Textual	146	5	Brand	48	0.33	0.2
Test set Numerical	75	3	Width	25	0.33	0.33

Table 4.30: Metadata results for the prediction experiment with bearingsonline in the test set and all other data sources in the training set.

	Nearest Neighbour	RbfSCV
Textual Score	0.55	0.69
Textual Standard Deviation	0.020	0.057
Numerical Score	0.79	0.55
Numerical Score Deviation	0.020	0.051

Table 4.31: Results for the prediction experiment with Bearingsonline in the test set and all other data sources in the training set.

	Sample Number	Attribute Count	Most Common Attribute	Most Common Attribute Count	ZeroR	Pure Random
Training set Textual	4989	27	Product ID	846	0.17	0.04
Training set Numerical	1266	6	Width	398	0.31	0.17
Test set Textual	146	5	Brand	48	0.33	0.2
Test set Numerical	75	3	Outside Diameter	25	0.33	0.33

Table 4.32: Metadata results for the prediction experiment with Bearingsonline in the test set and all other data sources in the training set. The data sources have been filtered such that only columns with a minimum of 750 data records are used.

	Nearest Neighbour	RbfSCV
Textual Score	0.35	0.62
Textual Standard Deviation	0.012	0.037
Numerical Score	0.76	0.91
Numerical Score Deviation	0.070	0.046

Table 4.33: Results for the prediction experiment with Bearingsonline in the test set and all other data sources in the training set. The data sources have been filtered such that only columns with a minimum of 750 data records are used.

	Sample Number	Attribute Count	Most Common Attribute	Most Common Attribute Count	ZeroR	Pure Random
Training set Textual	5028	29	Product ID	881	0.18	0.03
Training set Numerical	1340	9	Width	405	0.30	0.11
Test set Textual	385	9	Title	58	0.15	0.11
Test set Numerical	144	3	Width	48	0.33	0.33

Table 4.34: Metadata results for the prediction experiment with Bearingsdirect in the test set and all other data sources in the training set.

	Nearest Neighbour	RbfSCV
Textual Score	0.26	0.26
Textual Standard Deviation	0.0	0.0
Numerical Score	0.33	0.33
Numerical Score Deviation	0.016	0.007

Table 4.35: Results for the prediction experiment with Bearingsdirect in the test set and all other data sources in the training set.

	Sample Number	Attribute Count	Most Common Attribute	Most Common Attribute Count	ZeroR	Pure Random
Training set Textual	4882	29	Product ID	857	0.18	0.03
Training set Numerical	1265	9	Width	380	0.30	0.11
Test set Textual	531	10	Title	83	0.16	0.1
Test set Numerical	219	3	Outside Diameter	73	0.33	0.33

Table 4.36: Metadata results for the prediction experiment with Bearingsonline and Bearingsdirect in the test set and all other data sources in the training set.

	Nearest Neighbour	RbfSCV
Textual Score	0.34	0.22
Textual Standard Deviation	0.002	0.011
Numerical Score	0.55	0.40
Numerical Score Deviation	0.011	0.012

Table 4.37: Metadata results for the prediction experiment with Bearingsonline and Bearingsdirect in the test set and all other data sources in the training set.

	Sample Number	Attribute Count	Most Common Attribute	Most Common Attribute Count	ZeroR	Pure Random
Training set Textual	4619	23	Product ID	788	0.17	0.04
Training set Numerical	1122	6	Width	350	0.31	0.17
Test set Textual	516	9	Title	83	0.16	0.11
Test set Numerical	219	3	Width	73	0.33	0.33

Table 4.38: Metadata results for the prediction experiment with Bearingsonline and Bearingsdirect in the test set and all other data sources in the training set. The data sources have been filtered such that only columns with a minimum of 750 data records are used.

	Nearest Neighbour	RbfSCV
Textual Score	.0.19	0.21
Textual Standard Deviation	0.004	0.001
Numerical Score	0.49	0.53
Numerical Score Deviation	0.027	0.031

Table 4.39: Results for the prediction experiment with Bearingsonline and Bearingsdirect in the test set and all other data sources in the training set. The data sources have been filtered such that only columns with a minimum of 750 data records are used.

	Sample Number	Attribute Count	Most Common Attribute	Most Common Attribute Count	ZeroR	Pure Random
Training set Textual	5279	30	Product ID	921	0.17	0.03
Training set Numerical	1400	8	Width	435	0.31	0.125
Test set Textual	134	10	Seal Type	18	0.13	0.1
Test set Numerical	84	5	Inside Diameter	18	0.21	0.2

Table 4.40: Metadata results for the prediction experiment with BTShop in the test set and all other data sources in the training set.

	Nearest Neighbour	RbfSCV
Textual Score	0.35	0.35
Textual Standard Deviation	0.0	0.0
Numerical Score	0.79	0.57
Numerical Score Deviation	0.039	0.020

Table 4.41: Results for the prediction experiment with BTShop in the test set and all other data sources in the training set.

	Sample Number	Attribute Count	Most Common Attribute	Most Common Attribute Count	ZeroR	Pure Random
Training set Textual	4145	31	Product ID	699	0.18	0.03
Training set Numerical	1189	8	Width	380	0.32	0.13
Test set Textual	1686	15	Product ID	240	0.19	0.07
Test set Numerical	295	4	Weight	77	0.26	0.25

Table 4.42: Metadata results for the prediction experiment with Eriks and ABF in the test set and all other data sources in the training set.

	Nearest Neighbour	RbfSCV
Textual Score	0.17	0.23
Textual Standard Deviation	0.011	0.009
Numerical Score	0.43	0.38
Numerical Score Deviation	0.030	0.013

Table 4.43: Results for the prediction experiment with Eriks and ABF in the test set and all other data sources in the training set.

	Sample Number	Attribute Count	Most Common Attribute	Most Common Attribute Count	ZeroR	Pure Random
Training set Textual	3861	26	Product ID	506	0.13	0.04
Training set Numerical	798	6	Cage Type	242	0.30	0.17
Test set Textual	1274	5	Brand	364	0.29	0.2
Test set Numerical	543	3	Width	181	0.33	0.33

Table 4.44: Metadata results for the prediction experiment with Xbearings in the test set and all other data sources in the training set. The data sources have been filtered such that only columns with a minimum of 750 data records are used.

	Nearest Neighbour	RbfSCV
Textual Score	0.34	0.41
Textual Standard Deviation	0.025	0.016
Numerical Score	0.90	0.97
Numerical Score Deviation	0.022	0.019

Table 4.45: Results for the prediction experiment with Xbearings in the test set and all other data sources in the training set. The data sources have been filtered such that only columns with a minimum of 750 data records are used.

	Sample Number	Attribute Count	Most Common Attribute	Most Common Attribute Count	ZeroR	Pure Random
Training set Textual	3241	27	Product ID	722	0.22	0.04
Training set Numerical	1188	6	Outside Diameter	357	0.30	0.17
Test set Textual	2143	20	Product ID	208	0.10	0.05
Test set Numerical	296	4	Product Series	99	0.33	0.25

Table 4.46: Metadata results for the prediction experiment with Motionindustries in the test set and all other data sources in the training set.

	Nearest Neighbour	RbfSCV
Textual Score	0.17	0.14
Textual Standard Deviation	0.008	0.021
Numerical Score	0.25	0.25
Numerical Score Deviation	0.027	0.018

Table 4.47: Results for the prediction experiment with Motionindustries in the test set and all other data sources in the training set.

	Sample Number	Attribute Count	Most Common Attribute	Most Common Attribute Count	ZeroR	Pure Random
Training set Textual	1582	22	Product ID	300	0.19	0.05
Training set Numerical	501	6	Width	128	0.26	0.17
Test set Textual	1659	11	Product ID	422	0.25	0.09
Test set Numerical	687	3	Inside Diameter	229	0.33	0.33

Table 4.48: Metadata results for the prediction experiment with Xbearings and Bearingsdirect in the test set and all other data sources excluding Motionindustries in the training set.

	Nearest Neighbour	RbfSCV
Textual Score	0.29	0.27
Textual Standard Deviation	0.006	0.022
Numerical Score	0.81	0.48
Numerical Score Deviation	0.010	0.018

Table 4.49: Results for the prediction experiment with Xbearings and Bearingsdirect in the test set and all other data sources excluding Motionindustries in the training set.

	Sample Number	Attribute Count	Most Common Attribute	Most Common Attribute Count	ZeroR	Pure Random
Training set Textual	681	6	Product ID	300	0.44	0.17
Training set Numerical	293	3	Width	98	0.33	0.33
Test set Textual	1327	6	Product ID	422	0.32	0.17
Test set Numerical	687	3	Outside Diameter	229	0.33	0.33

Table 4.50: Metadata results for the prediction experiment with Xbearings and Bearingsdirect in the test set and all other data sources excluding Motionindustries in the training set. Data filtered such that only columns associated to attributes that are present in both test set and training set are used.

	Nearest Neighbour	RbfSCV
Textual Score	0.63	0.63
Textual Standard Deviation	0.001	0.001
Numerical Score	0.81	0.84
Numerical Score Deviation	0.000	0.000

Table 4.51: Results for the prediction experiment with Xbearings and Bearingsdirect in the test set and all other data sources excluding Motionindustries in the training set. Data filtered such that only columns associated to attributes that are present in both test set and training set are used.

Experiment Discussion: It is difficult for us to draw any final conclusions due to the wide variance of results between the different training/test splits. We have some splits where we get good results and some where we barely score better than a ZeroR method. Overall it seems that there is still potential for our methodology given some of our results. In our current implementation our machine learner is not capable of recognising new attributes meaning that attributes which are not present in the training set but are in the test set are guaranteed to be evaluated incorrectly. Going by the differences between our last 2 experiments where we tested ensuring that only those attributes present in both were used indicates to us that at least one good improvement would be adding the ability to recognise new previously unknown attributes. Assessing the two machine learning algorithms, we feel that the Nearest neighbour algorithm performs best when considering numerical cases while textual data seems better handled by the RbfSVC algorithm. There are however still exceptions to this and these score differences could be caused by our case where on average we have a smaller number of attributes for numerical data than textual and it is this difference in number of attributes that causes the performance difference and not anything inherent in the data types. An interesting aspect about our final experiment is that our test set is actually significantly larger than our

training set but it seems that despite this size difference both algorithms score well in this case.

4.4 MacroScale Experiment

Hypothesis: Our overall end goal is to match columns to columns by way of their attributes. We have gotten difficult to asses results from our prediction experiment but by moving our prediction experiment to a macro scale we may get some good results.

Experiment Setup: We split our full Deep Grooved Ball Bearing dataset into a training set and a test set. We will do this on a per data source level. We process each data set separately using our preprocessing steps. We create samples of 100 data value size and use the resampling with limiters method while attempting to generate 1000 samples for each column. We will generate all features. We transform the generated features to Unit Standard Deviation. In the case of the test set we use the transformation function based on the training set. We use the processed training set data to train two machine learners using gridsearch. One machine learner uses the Nearest neighbour algorithm with hyperparameters for each uneven amount of neighbours from 1 to 31 and using .euclidean or minkowski distance metrics. The other uses RbfSVC with a C of 10-3 through 103, increasing in steps of 1 exponent as its hyperparameter. Using the predict function from each of the classifiers we predict the test set.

Variations in the Experiment: Xbearings and Bearingsdirect in the test set with all other data sources except Motionindustries in the training set. Data filtered such that only columns associated to attributes that are present in both test set and training set are used. Data is also filtered to only include columns which have a minimum size of 750 data values.

Measurements: For assessing the results of this experiment we will perform the experiment 5 times. We record the results on a column by column basis. For each column is 51% or more of its samples are predicted correctly we consider the column as a whole to have been predicted correctly but if this threshold is not reached than the whole column is considered to have failed. We also compute an overall score which is the mean success rate of the columns with a success being represented by a 1 and a failure by a 0. To gain an insight into the data we are working with we also collect a number of metadata values. The meta data collected are:

- The number of Samples
- Attribute Count
- Most Common Attribute
- Most Common Attribute Count in samples
- ZeroR Score

- Pure Random Score

Experiment Results:

	Sample Number	Attribute Count	Most Common Attribute	Most Common Attribute Count	ZeroR	Pure Random
Training set Textual	610	6	Product ID	282	0.46	0.17
Training set Numerical	293	3	Width	98	0.33	0.33
Test set Textual	1327	6	Product ID	422	0.32	0.17
Test set Numerical	687	3	Inside Diameter	229	0.33	0.33

Table 4.52: Metadata results for the macro scale prediction experiment.

	Nearest Neighbour	RbfSCV
Textual Score	0.55	0.53
Textual Standard Deviation	0.0	0.036
Numerical Score	0.67	0.67
Numerical Score Deviation	0.010	0.018

Table 4.53: Results for the macro scale prediction experiment.

Textual		Numerical	
xbearings_Brand	1.0	xbearings_Inner Diameter (d)	0.98
xbearings_Product	1.0	xbearings_Outer Diameter (D)	1.0
xbearings_Product model	0.0	xbearings_Thickness (B)	0.91
xbearings_New Bearing model	1.0	bearingsdirect_ID	0.88
xbearings_Types	0.0	bearingsdirect_OD	0.0
xbearings_Brands	1.0	bearingsdirect_W	0.15
bearingsdirect_Title	0.17		
bearingsdirect_Product	1.0		
bearingsdirect_LUBRICATION	0.02		
bearingsdirect_Description	1.0		
bearingsdirect_BRAND	0.0		

Table 4.54: By column results for the macro scale prediction experiment for the nearestNeighbour algorithm.

Textual		Numerical	
xbearings_Brand	1.0	xbearings_Inner Diameter (d)	0.96
xbearings_Product	1.0	xbearings_Outer Diameter (D)	1.0
xbearings_Product model	0.0	xbearings_Thickness (B)	0.93
xbearings_New Bearing model	1.0	bearingsdirect_ID	0.88
xbearings_Types	0.0	bearingsdirect_OD	0.0
xbearings_Brands	1.0	bearingsdirect_W	0.10
bearingsdirect_Title	0.03		
bearingsdirect_Product	1.0		
bearingsdirect_LUBRICATION	0.0		
bearingsdirect_Description	1.0		
bearingsdirect_BRAND	0.0		

Table 4.55: By column results for the macro scale prediction experiment for the RbfSVC algorithm.

Experiment Discussion: Overall both algorithms score fairly acceptable and similarly. When looking on a column by column basis this remains true but it is very interesting to see that unlike our expectations a lot of columns have a success rate of either 100% or 0%.

Examining the raw data shows that for at least some of the properties they do indeed seem to differ from the general appearance. The Brand property from BearingsDirect for example seems to differ significantly from the general property. In general brand is denoted with a single short word but in the case of Bearings Direct it is often multiple words and significantly longer. Lubrication is another property our macro prediction experiment scores poor on. Examining this property in more detail shows that only one other source (ABF) also has lubrication as a property but how they are used in different. In the ABF data source a general lubrication type is described, while in bearingsdirect the lubrication varies between a specifically branded lubricant or oil of various types.

4.5 Summary

On the basis of the experiments executed in this chapter we have refined our method and done some evaluation about the performance of our method in a variety of circumstances.

- On the basis of a large number of experiments we are able to conclude that a resampling algorithm with a clear minimum size requirement and a maximum number of samples that can be generated from a single column combined with a sample size of 100 and intending to generate a maximum of 1000 samples from each column gives us good results. (4.1)
  - The minimums and maximums are important to use to prevent a resampling algorithm from generating too many samples from too

small a column which results in overtraining from too many duplicates. (4.1.5-4.1.6)

- From our experiments into which machine learning algorithms perform best we were able to conclude that both NearestNeighbour and a Rbf-based SVC are both excellent classification algorithms to use for our method. Both these algorithms score well in their selection experiments (98%+ for textual and 91%+ for numerical) and continue to score well in the prediction experiments. (4.2)

- Under the circumstances that we split our feature sets between test and training with no discernment about data source or column origins, the algorithm get excellent accuracy scores (90%+). This indicates that our theory of matching on the basis of data characteristics and commonality is usable.

- When the feature sets are split between test and training on a data source level and use the predict function, the algorithm has a larger variance in performance. (4.3-4.4)

  - If only feature sets associated with properties that are present in both the test and training set are used and all other feature sets are discarded, the algorithm scores very well. (4.3 tables 4.48-4.51 and 4.4)

    * Under analysis of a macro test where each feature set is grouped by column it seems that our method is very polarized in its accuracy with columns being either in the region of 100% accurate or 0% accurate. (4.4)

  - We have analysed cases such as BearingsOnline in the test set with the columns filtered to a minimum size of 750 data values where we have a matching accuracy of 62% on textual data and 91% accuracy on numerical data for the best performing algorithm. (table 4.32 and 4.33)

  - But we also found cases such as MotionIndustries being the data source which scores 17% and 25% on the textual and numerical accuracy respectively using its best performing algorithm. (table 4.42 and 4.33)

  - Within our data set numerically typed properties are in general easier to match than textual data. This seems logical given the following circumstances

    * Within our data set the number of numerical properties is significantly lower than the number of textual properties and thus would be easier to match given that there is less choice.

    * There is less freedom in how to represent numerical data since in the end it comes down to a number.

- We see a number of other causes for the variance in performance and our varying results are explainable by them as in how much each data selection set suffers from these.
    * Our current machine learner implementation is not able to handle (and automatically separate or exclude) unknown properties in the test set, therefore always classifying those incorrectly.
    * Due to the freedom within natural language and how data can be represented differently within various data sources it can occur that columns which represent the same property are not comparable to each other without additional work which is not always feasible.

Chapter 5

Conclusion

In this thesis we present a method for the automatic matching of properties across different data sources on the basis of data characteristics and commonality. The results from this method can then be used to generate database matching schemas to allow for the merging of data sources. Our method works by separating each property within each data source out into their own columns. These separated columns are then further divided into smaller samples from which features that represent data characteristics of those columns are calculated. By using a machine learner with these features a classifier is built capable of matching the properties of the same meaning together.

Our main research question was also formulated to reflect this:
"How can we link properties which represent the same real world property or concept together, using the characteristics and commonalities of their data."

For our method we used a number of machine learner usable features to represent data characteristics. By using these features we are able to match columns to attributes by use of a classification based machine learner. The features we calculate are based on aggregated data from a single column and are different depending on data type.

Six features are used for numerical data:

- The value of the most common entry within the sample
- The number of unique entries represented as a percentage to avoid the sample size from influencing this.
- The minimum numerical value within the sample
- The maximum numerical value within the sample
- The mean of all numerical values within the sample

- Standard Deviation of all numerical values (from mean) within the sample

Nine features are used for textual data:

- The number of unique entries represented as a percentage to avoid the sample size from influencing this.
- The minimum number of characters in a value within the sample
- The maximum number of characters in a value within the sample
- The mean number of characters within the sample
- Standard Deviation within the sample in terms of number of characters
- The minimum number of words in a value within the sample
- The maximum number of words in a value within the sample
- The mean number of words within the sample
- Standard Deviation within the sample in terms of number of words

It is important to have a sufficient number of samples available for the machine learner based classification to work with. To increase the number of samples that can be generated from a data source we use resampling techniques. It is however important to prevent overtraining from happening so we use a resampling technique with limiters on how many samples can be generated. The experiments show best performance with a setting in which samples are generated at size of 100 values, where samples can only be generated from a minimum column size of 200, and in total no more samples can be generated than 3 times the number of data values within the column.

Our experiments show that of standard classification algorithms K-NearestNeighbour and a Support Vector Machine with a Rbf-based kernel show good matching results.

In the end our final prediction experiments result in somewhat mixed outcomes. We have several cases where the algorithm has excellent accuracy scores given the number of involved properties, but also some cases with lower scores. We have found two factors which seem to be significant contributors in our varying scores.

- The current system is not yet able to recognize new properties in the test set and automatically assign a new class to them. These new properties can of course never be predicted correctly resulting in a lower overall accuracy score in cases where they are present.
  - In the case where we filter our experiment data to not contain such new properties, asserting that only properties that are present in both training and test set are used, we have great accuracy. Suggesting that the method as such is a good tool for database schema matching.

  - Another case where the algorithm scores well is when using BearingsOnline as the test set and all other data sources as the training set. The test set in this case also does not contain many properties unknown within the training set.

- Our assumption of properties having similar characteristics holds in the majority of columns in our data set. However we still have some columns within the dataset where this is unfortunately not the case. This is more common for textual data than for numerical values. Due to the freedom within natural language and how data can be represented differently across different data sources, such difficulties will always remain. It seems to be the case that in general some standards are used across a domain resulting in this issue only occurring for a low number of cases.

  - An example of this would be the brand column of BearingsDirect which in our macro scale test had poor matching results since it represents brands differently than the general case by including additional elements into the brand names such as Ltd., Co., and country of manufacturing.

Overall we can say that our developed method is suitable for matching tasks where most of the properties to be matched are already present in the training data and with more work done such as described in the future work section could be made suitable for an expanded range of matching tasks.

Chapter 6

Future Work

During the development of our preprocessing steps and the execution of our experiments we have found a number of areas where future work could be done.

- The most likely single method to yield significant further improvement is the ability for the machine learner to recognise and deal with unknown properties. As we found in our prediction experiments there is a large variance in how several different test sets perform. A primary cause of this is that some test sets contain unknown properties compared to the training set. As our machine learner does not possess the capacity to deal with unknown properties these will never be predicted correctly. Our machine learner can only classify properties to those it has learned from the training set and does not possess the capability to detect unknown properties and classify them to a seperate class or classes in the case of several distinct unknown properties. Our experiments in which we only had known attributes in both the test and training set (thereby eliminating the deteriorating effect of unknown attributes) showed a significant improvement over the same test and training set where all attributes were included.

- During this thesis a large amount of our focus has been on matching on a sample by sample level as this is the level on which the actual matching takes place. Our system provides a way to abstract this to a column by column scale. Further research and development into matching on a higher scale would be beneficial from a practical perspective. As it currently stands we have a number of features which we use to classify, however the selection of features could benefit from additional experimentation and research.

- Our current preprocessing system has been designed for this specific data set. If it is desired to implement this method for general use the preprocessing system should be changed to accommodate more generic use or an

analogous system which is also capable of generating the required input for the machine learning phase.

- The number of data types could be expanded.
- Our machine learner classification has been mostly used with the aim of experimentation and validation of our method. When aiming to turn the overall system for general use, expanding the machine learner feedback can be valuable. Types of feedback include providing a top 3 classes for every match and confidence numbers.

Finally our experiments seem to indicate that at within our data set and perhaps generalizable to product data that data values within a column are similar to each other. Our research is about the data similarity across multiple data sources but research into exploiting this similarity within a single data source could be an interesting avenue of research. An example of this is a method to detect data aberrations.

Chapter 7

Acknowledgements

We thank Maurice van Keulen, Dolf Trieschnigg, and Doina Bucur for the guidance and supervision provided for this project. We would also like to thank Lilian Spijker and Suze Engbers for their support.

Bibliography

[1] Jens Bleiholder and Felix Naumann. Data fusion. *ACM Computing Surveys*, 41(1):1:1–1:41, January 2009.

[2] Xiaoyi Wang. Matching records in multiple databases using a hybridization of several technologies. 2008.

[3] John M. Smith, Philip A. Bernstein, Umeshwar Dayal, Nathan Goodman, Terry Landers, Ken W. T. Lin, and Eugene Wong. Multibase: Integrating heterogeneous distributed database systems. In *Proceedings of the May 4-7, 1981, National Computer Conference*, pages 487–499. ACM, 1981.

[4] Indrajit Bhattacharya and Lise Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):5, 2007.

[5] AnHai Doan, Pedro Domingos, Pedro Domingos, and Alon Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. *SIGMOD Record*, 30(2):509–520, May 2001.

[6] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, Dec 2001.

[7] Philip A. Bernstein, Jayant Madhavan, and Erhard Rahm. Generic schema matching, ten years later. *Proceedings of the VLDB Endowment*, 4(11):695–701, 2011.

[8] Tanvi Sahay. Schema matching using machine learning. `https://tanvisahay.github.io/Schema_Matching.pdf`, 2017.

[9] Amihai Motro, Jacob Berlin, and Philipp Anokhin. Multiplex, fusionplex and autoplex: Three generations of information integration. *SIGMOD Record*, 33(4):51–57, December 2004.

[10] Jennifer Rowley. The wisdom hierarchy: representations of the dikw hierarchy. *Journal of Information Science*, 33(2):163–180, 2007.

[11] Alan M. Turing. Computing machinery and intelligence-am turing. *Mind*, 59(236):433, 1950.

[12] Arthur L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

[13] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2009.

[14] Yves Kodratoff. *Introduction to machine learning*. Elsevier, 2014.

[15] Sotiris B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24, 2007.

[16] Colin Conrad, Naureen Ali, Vlado Keelj, and Qigang Gao. Elm: An extended logic matching method on record linkage analysis of disparate databases for profiling data mining. In *2016 IEEE 18th Conference on Business Informatics (CBI)*, volume 01, pages 1–6, Aug 2016.

[17] Tim Churches, Peter Christen, Kim Lim, and Justin X. Zhu. Preparation of name and address data for record linkage using hidden markov models. *BMC Medical Informatics and Decision Making*, 2(1):9, Dec 2002.

[18] William W. Cohen and Haym Hirsh. Joins that generalize: Text classification using whirl. In *Knowledge Discovery and Data Mining*, volume 98, pages 169–173, 1998.

[19] Bureau International des Poids et Mesures. Text of the resolutions adopted by the 22nd general conference on weights and measures. page 11, 2003.