UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

COMPUTER SCIENCE HONOURS
FINAL PAPER
2015

Title: Transforming DSpace into a Research Information Management System: Automatic and Manual Metadata Mapper

Author: Craig Feldman

Project Abbreviation: NRFDB

Supervisor: Associate Professor Hussein Suleman

Category	Min	Max	Chosen
Requirement Analysis and Design	0	20	0
Theoretical Analysis	0	25	0
Experiment Design and Execution	0	20	18
System Development and Implementation	0	15	12
Results, Findings and Conclusion	10	20	15
Aim Formulation and Background Work	10	15	15
Quality of Paper Writing and Presentation	10		10
Adherence to Project Proposal and Quality of Deliverables	10		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	0	10	0
Total marks		**80**	

Transforming DSpace into a Research Information Management System: Automatic and Manual Metadata Mapper

Craig Feldman
University of Cape Town, South Africa
fldcra001@myuct.ac.za

ABSTRACT
This paper presents the development of an automatic and manual metadata mapper for DSpace. The idea for the development of this tool arose from a request by the National Research Foundation of South Africa to migrate data into a DSpace repository. We identified that there was a need for an easy to use means to import data and map it to the appropriate metadata fields as used by DSpace. The tool developed and discussed in this paper aims to facilitate the deposit and migration of data into a DSpace repository. Machine learning is used to attempt to determine the appropriate Dublin Core metadata field to which each field of the input data belongs. Various algorithms were tested and, through cross-validation, it was found that Random Forest was the best performing algorithm. Usability testing showed that the system developed provided an effective and usable means to import data into a DSpace repository.

CCS Concepts
• Applied computing → Digital libraries and archives

Keywords
Institutional repositories, DSpace, database migration, textual classification.

1. INTRODUCTION
The formulation of this project arose from a request by the National Research Foundation (NRF) of South Africa to assist them in migrating data from a legacy database system into the more modern and sophisticated DSpace[1] digital repository system. After performing research into the needs of not only the NRF, but the research management community as a whole, we set out to create a set of tools with the aim of transforming DSpace into a Research Information Management System (RIMS)[2].

This paper presents information relating to an add-on that was developed for DSpace – an automatic and manual metadata mapper. This tool aims to assist users when adding data to DSpace by attempting to automatically map the fields of the legacy system to that of the metadata fields used by DSpace. Machine learning was used to try to predict which Dublin Core metadata field a given entry should be classified as. Five different machine learning algorithms were selected and compared to determine the best performing algorithm for this task. This tool also allows for data to be added to an existing DSpace repository, and the metadata mappings can be saved for future use. It is accessed through a Web-based user interface that also allows for the user to review and correct the attempted automatic metadata mappings.

1.1 Project Significance
According to the Registry of Open Access Repositories (ROAR)[3], DSpace is the most widely used digital repository system in the world, therefore useful additions to DSpace are likely to be well received. The current solutions for migrating legacy data into DSpace involves users having to understand how the legacy system stored the data, then creating custom scripts capable of formatting the data into a DSpace accepted format, as well as specifying how the data maps to the DSpace metadata fields. The metadata mapper aims to make this process easier, faster and more user friendly. It is hoped that this software will help encourage users to transfer their research management system into the modern and feature rich DSpace. This would be beneficial not only to the people responsible for the management of that research information, but also to the general research community as research can be more effectively preserved, managed and distributed.

This tool was developed along with two other tools that were developed by Darryl Meyer – a report writer and ingestion manager for DSpace. The report writer is not discussed in this paper, but certain indirect references are made to the ingestion manager which is used by the metadata mapper to add items into a DSpace repository. It is hoped that these tools will help transform DSpace into a RIMS by adding features to DSpace that are already available in RIMS software packages.

1.2 Project Aims
This project had the primary aim of developing tools that will facilitate the migration of data, from a legacy database system, into DSpace. An initial survey was sent to a number of DSpace mailing lists, asking for input into this project. The questions and results of this survey are provided in the appendix. One issue that was identified from this survey is that mapping the fields from the source to the Dublin Core metadata fields was time-consuming and there is no easy and intuitive way of doing so. Thus we set out to investigate whether we could develop a tool that offered a more usable and easier to use interface than that which is currently offered by DSpace. This tool would accept a CSV file containing the source data and use machine learning techniques to try to automatically identify to which Dublin Core field each entry belongs.

Secondary to this, we aimed to use these tools to assist with the database migration for the NRF and hoped that these tools would prove useful for them.

[1] See section 2.1
[2] A RIMS is used to store and manage the intellectual data created by an institution.
[3] http://roar.eprints.org/

This paper references supplementary material that can be accessed via UCT's Department of Computer Science online publications page at: http://pubs.cs.uct.ac.za. The project abbreviation used for this paper is 'NRFDB'. All references to appendices made in this paper refer to these online supplementary materials.

1.3 Structure of Report

This report first introduces a number of relevant papers and projects as well as some of the latest state of the art work in the field of metadata mapping and database migrations. Thereafter, the report is broadly split into two sections – the first deals with the development and execution of an experiment to determine the best machine learning algorithms for the given problem, the results of which are discussed in section 4. Section 5 then presents information relating to the software development process and user interface (UI) design. The results of usability testing, as well as feedback received from the NRF, are discussed in section 6. Finally, the report presents the conclusions that were reached, as well as areas for future work.

2. BACKGROUND

2.1 DSpace

According to Smith et al. [31], the development of DSpace's architecture drew on the information provided in Kahn & Wilensky's [14] Framework for Distributed Digital Object Services (which aimed to describe fundamental aspects of an open architecture infrastructure that supports systems such as digital libraries), as well as Arms' [1] work on Key Concepts in the Architecture of the Digital Library.

DSpace was developed by MIT to address an issue faced by their library of having to collect, preserve, index and distribute an increasing number of scholarly publications and research materials, presented in complex digital formats; this was both time-consuming and costly [31]. DSpace aims to act as a repository to give digital research and educational material greater visibility and accessibility over time. It was created to address all the basic functionality required in a digital repository service, with the intention of being expanded upon in the future, particularly to address long term data preservation concerns [33]. According to the DSpace website[4], some of the main reasons to use DSpace are that it:

- Can be customised to fit the institution's needs.
- Can be easily installed and configured.
- Can manage and preserve all types of digital content.

DSpace uses a qualified Dublin Core metadata standard for describing items [31]. There are 15 main Dublin Core fields, and each field may contain several qualifiers. A list of the default DSpace 15 Dublin Core fields, along with their qualifiers, can be found in the appendix.

2.2 Database Migration

The complexities of the NRF database migration arise from the need to transition an old legacy system to a new modern system (DSpace). As such, it is important to focus on similar projects to investigate past experiences and practices. According to Bisbal et al. [4], legacy systems can pose considerable problems, including brittleness, inflexibility, non-extensibility and a lack of openness.

2.2.1 General Principles of Data Migration

Bisbal et al. [4] argue that the naïve approach to migrating a legacy system involves redeveloping the system from scratch using modern tools, however, the risk of failure is usually very high when using this approach. Instead, Brodie & Stonebraker [6] suggest three different approaches. (1) The Forward Migration Method which involves first transferring the legacy data to the new, modern database system and then incrementally migrating the legacy applications. (2) The Reverse Migration method where the legacy applications and interfaces are migrated, followed by the data. (3) The Composite Database approach whereby legacy applications are gradually rebuilt and the legacy and target system form a composite system during migration. Wu et al. [40] proposed the 'Butterfly Methodology' as an alternative to the current thinking on legacy system migrations. The Butterfly Methodology eliminates the need for users to simultaneously access both the legacy and target systems by dividing the system migration into six independent and sequential migration activities.

2.2.2 DSpace Data Migration

The primary means of adding items into DSpace are to either enter each entry via the DSpace Web portal or in batch upload via the DSpace item importer (a command-line tool for batch ingesting items that makes use of a simple archive format) but enhancements to DSpace include new deposit options making use of SWORD[5], OAI-ORE[6], and DSpace package importers [37]. Many projects have been implemented that make use of scripts to automate the process of creating the archive directory to assist in batch uploading. There is a considerable amount of literature documenting the methods used for batch ingestion to populate institutional repositories. Mishra et al. [21] and Mundle [22] developed Perl scripts to create the DSpace archive directory for batch imports of electronic theses and dissertations (ETDs) whereas Brownlee [8] made use of Python scripts to process CSV files (created using FileMaker[7]). Walsh [37] describes using Perl scripts to migrate data from spreadsheets and CSV files into the DSpace archive format for the Ohio State University's institutional repository. Ribaric [26] describes the use of PHP utilities for the automatic preparation of ETDs (from the Internet Archive[8]) for deposit into DSpace. Several other similar projects are introduced by Walsh [37].

From the above, it is clear that a considerable number of database migrations into the DSpace platform led to customized 'throw away' tools being developed to migrate data from a legacy system into DSpace. There appears to be a need for a generic tool that would take a standardized file as input, and automatically add this to a DSpace repository. Such a tool could also attempt to perform automatic matching of fields from the input data to those of the DSpace Dublin Core metadata fields, thus further simplifying the process of migrating a legacy system to DSpace.

2.3 Metadata Mapping

The process of mapping legacy data fields to metadata fields from the qualified Dublin Core metadata standard can be seen as being strongly related to a process known as schema mapping (mapping a source database into a different, but fixed, target schema) [20].

2.3.1 General Principles and Potential Issues

Haslhofer & Klas [11] looked at various techniques to achieve interoperability amongst metadata standards by (1) agreeing on certain metadata models, (2) agreeing on a common meta-model by

[4] http://www.dspace.org/

[5] A protocol for depositing content from one location to another - http://swordapp.org/about/

[6] A set of standards for the description and exchange of aggregations of Web resources - https://www.openarchives.org/ore/

[7] http://www.filemaker.com

[8] http://www.archive.org

forming a relationship between models to a common meta-model, or (3) reconciliation of the structural and semantic heterogeneities. In their paper, they also discuss how various problems can arise when moving between metadata standards or trying to map fields to a metadata standard from a legacy system. Some of these issues have already been identified in the early history of database research, with the first in depth analysis of this field being conducted by Sheth & Larson [29]. These issues have been investigated in more detail throughout the years (for example by Ouksel & Sheth [23], Visser et al. [35] and Wache [36]). Haslhofer & Klas [11] found that a number of problems may inhibit the interoperability of different metadata standards and fields - these include either structural or semantic differences.

Structural differences in the metadata models may include things such as conflicting constraints placed on the metadata fields, differences in field names, as well as more complicated problems such as those arising from implicit or explicit generalization and aggregation in the metadata fields. Various other structural issues are discussed in the paper by Haslhofer & Klas [11].

Semantic issues occur as a result of differences in the semantics of the metadata models. Semantic issues include when the domains of the metadata standards differ and cannot be mapped to one another and when different units are used to represent a field (for example, when one metadata standard represents a width in pixels, and another in centimeters, or when different date formats are used).

2.3.2 Prior Research and Approaches

Various techniques and approaches have been used to perform metadata mappings and database migrations. The techniques and research that have been employed by past projects is discussed here.

Rahm & Bernstein [25] present a number of papers that make use of machine learning to learn how fields map between databases, for example the LSD (Learning Source Descriptions) system that uses machine learning techniques to semi-automatically create semantic mappings [9, 10]. The system has two general phases: training and matching. In the training phase, the user is required to specify a one-to-one mapping from several input sources. This supervised learning approach is used to extract different types of information from the source schema and data to train a set of 'base learners'. These base learners can be any set of algorithms that accept inputs and produce an output, for example, neural networks or Bayesian based algorithms. The outputs of these base learners are then used to train a 'meta-learner' that uses a technique called stacking [34, 39] to combine the predictions of the base learners into one output. Once the training phase is complete, LSD can be used to predict the semantic mapping for unseen, new sources. The LSD system is shown in Figure 1.

Artificial Neural Networks (ANNs) have been used as an effective tool in schema mapping. For example, Li & Clifton [18] present a procedure for using an ANN to classify attributes based on their field specifications and data values. Li et al. [17] applied ANNs to the problem of mapping corresponding attributes between two databases. In their paper, they describe how they used an ANN in a database integration problem and how they represented an attribute with its metadata as discriminators (inputs to the ANN). They focus on experiments into the effectiveness of ANNs and each input, as well as the difficulties involved in using ANNs for this problem. The authors were satisfied with the ANN's performance in general but felt that the results could be further improved by incorporating

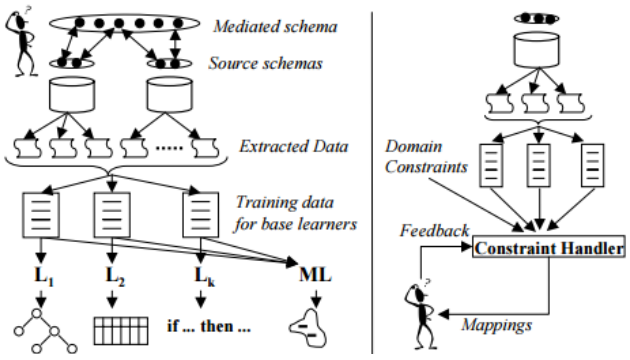

Figure 1: The two phases of the LSD system [9]

methods using other information such as attribute names. Many of the problems they faced are those discussed in section 2.3.1.

The SEMINT system [16] uses a neural network to match schema elements using properties such as field specifications (e.g. data types and sizes) as well as statistics based on the data content (e.g. maximum, minimum and mean values). Unlike the LSD system however, it does not exploit other types of data information such as word frequencies and field formats.

Berlin & Motro [3] describe a system called Automatch that uses machine learning techniques to automate schema mapping. The system described is based primarily on Bayesian learning and acquires probabilistic knowledge from examples that have been provided by domain experts. The system was able to show performance that exceeds 70% (measured as the harmonic mean of precision and recall).

Miller et al. [20] developed a semi-automatic tool that tried to help solve the schema mapping problem. The tool developed, Clio, employs a 'mapping-by-example' paradigm that allows the user to select mappings between fields based on example data provided to the user. Clio dynamically adjusts and ranks alternative mappings, with the top-ranked alternatives being suggested first.

Witten et al. [38] developed a tool to allow users to easily migrate data between DSpace and Greenstone[9] (another tool for managing digital libraries). In order to map the metadata from Greenstone, they developed a 'metadata crosswalk' that specifies the source and metadata elements and how they map to their Dublin Core counterpart.

Shvaiko & Euzenat [30] look at numerous different state of the art matching solutions and introduce their own technique for schema matching that they say "builds on top of state of the art in both schema and ontology matching". The authors build on top of previous state of the art solutions and add innovations such as introducing new criteria which are based on general properties of matching techniques, as well as interpreting the input information. In their research, they found that the state of the art matching systems tend to incorporate various techniques into a single 'hybrid approach'. For example, the 'Cupid' system that makes use of a hybrid matching algorithm that combines both linguistic and structural schema matching techniques to compute a similarity index [19].

[9] http://www.greenstone.org/

3. EXPERIMENT DESIGN AND EXECUTION

In order to test which machine learning algorithm was most effective at performing the required tasks, it was necessary to develop a set of experiments.

3.1 Overview of Experiment

In order to compare the performance of the chosen machine learning algorithms, Weka[10] was used to train and test the selected algorithms. Training data was gathered from a variety of open access repositories and a simple Java application was then written to extract numerical features from this training data and generate the input file that is used by Weka for training and testing. Cross-validation, as well as an unseen data set were used to compare the performance of the various algorithms. This process is discussed in more detail in the following sections.

3.2 Machine Learning Framework

Weka was identified as a good machine learning framework to use as it provides a large collection of machine learning algorithms as well as the ability to perform statistical tests on the comparative performance of the algorithms.

The aim of this experiment was to identify the most effective machine learning algorithm to use in the production software. Fortunately, Weka provides an 'Experimenter' mode which allows the user to select various machine learning algorithms to be trained and tested through cross-validation[11]. It then outputs results for each algorithm (such as the average percentage of correctly classified instances) and uses statistical tests to compare the algorithms' performance.

3.3 Training Data

As with any supervised machine learning algorithm, it is important to ensure that the training data is of a high quality and that it provides a representative sample of the inputs that are likely to be used in the future.

3.3.1 Gathering Training Data

In order to ensure that the training data would be effective in classifying unseen data, diverse data from a variety of sources was gathered. It was hypothesized that different repository software, as well as different metadata standards, would tend to each have their own distinct bias in how metadata is produced as different metadata standards may require different (or differently formatted) values (for example, different date formats). This would prove useful in gathering representative and diverse data. It was also important that each Dublin Core field was represented by the training data. Thus, OpenDOAR[12] and ROAR[13] were used to find a number of repositories from which the training data could be harvested through OAI-PMH[14]. It is also possible to filter the repositories by software (e.g. DSpace, Fedora[15], EPrints[16] etc.). Repositories from specific software were then selected in proportion to the ubiquity of that repository software and care was taken to ensure that the selected repositories contained high quality metadata. The metadata that was harvested from these repositories included a number of different metadata formats such as Dublin Core and ETD-MS[17]. A total of 32 813 training instances were gathered from 10 different sources. The distribution of these training instances can be found in Table 1.

Table 1: Distribution of Dublin Core fields for each test set

Dublin Core Field	Training Data	NRF Test Set
Contributor	206	1796
Coverage	103	0
Creator	11670	13380
Date	2199	16667
Description	2133	29524
Format	1454	0
Identifier	2434	0
Language	1246	13381
Publisher	854	0
Relation	80	0
Rights	1015	0
Source	629	0
Subject	6479	21729
Title	1681	13516
Type	630	0
Total	32813	109993

3.3.2 Filtering and Classifying the Training Data

The integrity of the training data was analysed and any 'outliers', such as incorrectly labelled data, were removed. One potential issue was that data from specific sources may over-represent a specific Dublin Core field and bias the machine learning algorithm towards that training set (overfitting). Such problems are however more prevalent in algorithms such as Artificial Neural Networks as opposed to decision tree based algorithms (by the nature of these algorithms) and this can often be avoided by fine tuning the algorithms' parameters. For example, Sebastiani [28] suggests that trees be 'pruned' to avoid overfitting.

The data was then labelled and classified into the 15 primary Dublin Core fields. The qualifiers were purposefully ignored, as it would lead to further complexities in classifying input data and would not likely yield accurate results.

3.3.3 Feature Extraction

Once the training data had been effectively classified, it was necessary to create a 'feature vector' – a numerical list of attributes that represent the training example. These feature vectors could then be used by Weka as inputs to a classification system.

Feature extraction involves representing the data that needs to be classified as a list of numerical attributes. This is done because machine learning algorithms are not aware of the meaning of a word and hence we need to represent the string numerically and in a manner that can be interpreted and used by the algorithm. The following 8 features/attributes were used for classification:

[10] http://www.cs.waikato.ac.nz/ml/weka/

[11] For an unofficial list and description of the classifiers, see http://wiki.pentaho.com/display/DATAMINING/Classifiers

[12] Directory of Open Access Repositories - http://www.opendoar.org/

[13] Registry of Open Access Repositories - http://roar.eprints.org/

[14] A protocol that is used to collect the metadata descriptions of records in an archive via HTTP requests.

[15] https://getfedora.org/

[16] http://www.eprints.org/

[17] Electronic Thesis and Dissertation Metadata Standard - http://www.ndltd.org/standards/metadata

- Number of characters.
- Number of words.
- Number of month names appearing in the string.
- Number of person names appearing in the string.
- Percentage of digits.
- Percentage of letters.
- Percentage of non-alphanumeric characters (excluding whitespace).
- Percentage of capital letters.

The number of person names was calculated by looking if a word appeared in a list of 50 000 of the most popular names from a US census. This, along with the number of month names appearing in the string, would help with the classification of creators and dates respectively.

3.4 Selected Machine Learning Algorithms

Weka provides a number of different machine learning algorithms that would be applicable. Five algorithms were selected for evaluation based on popularity and past performance in similar tasks and the default Weka parameters for these algorithms were used for testing and training. These algorithms are discussed in more detail in the following sections.

3.4.1 Naïve Bayes

The naïve Bayes model is one of the oldest, simplest and most common Bayesian based models [27]. It aims to predict a class variable (C) from a set of attributes ($x_1, \dots, x_n$), by applying Bayes' theorem. The model is 'naïve' because it assumes that the attributes are conditionally independent of each other, given the class. By applying Bayes' theorem, we find that:

$$P(C|x_1, \dots, x_n) \propto P(C) \prod_{i=1}^{n} P(x_i|C)$$

Thus, a deterministic prediction can be obtained by choosing the class with the highest relative probability [27].

3.4.2 Artificial Neural Networks

Artificial Neural Networks (ANNs) attempt to mimic the behaviour and structure of the brain by linking a set of artificial neurons via directed and weighted links [27]. The input attributes are fed through this network (the hidden layers), producing an output. Supervised learning occurs through a process called 'back-propagation' whereby training examples are fed through the network, with the error at the output layer being back-propagated through the hidden layers, and the weights being updated accordingly.

3.4.3 Logistic Regression

Unlike linear regression in which the dependant variable is categorical, logistic regression uses the same basic formula, but is modified to regress the probability of a categorical outcome [27]. Logistic regression is normally applied to a binary output but has been effectively extended to cases in which there are two or more possible discrete outcomes, as is the case here [13].

3.4.4 C4.5 Decision Trees

C4.5 is a popular and effective algorithm for generating decision trees. It was introduced in 1993 and has gained considerable popularity since ranking first in a paper presenting the top 10 data mining algorithms as identified by the 2006 IEEE International Conference on Data Mining [24, 41]. C4.5 builds a decision tree from a set of training data by using a divide and conquer technique based on splitting the tree by a test. Possible tests are ranked by the information gain (which minimises the total entropy of the subsets) and the default gain ratio (the ratio of information gain to the information provided by the test outcomes) [41]. Due to the popularity of this algorithm, it was chosen as the base algorithm against which the other four would be compared.

3.4.5 Random Forests

Random Forests are an ensemble learning method for classification that operates by constructing multiple decision trees during training. Each decision tree in the 'forest' is constructed on a random subsample of the training data and feature set [5]. Random forests are a popular tree-based classification technique as they correct for decision trees' habit of overfitting [12].

3.5 Testing of Machine Learning Algorithms

In order to test how well the selected algorithms would perform in the final system, it was necessary to carry out various tests on them. This is described further in the following sections.

3.5.1 K-Fold Cross-Validation

K-fold cross-validation is a statistical technique that involves dividing the training data into k randomly partitioned subsamples. A single subsample is retained for testing, and the remaining k-1 subsamples are used for training the algorithm. This process is repeated k times (the folds), where each subsample is used exactly once as the validation data. The results for each fold are then averaged. The default Weka parameter of 10 folds was used, as this value has experimental support as provided by Kohavi [15]. Cross-validation allows for algorithms to be effectively tested and compared based on how they are likely to perform for unseen data. Cross-validation was done using the training data as discussed in section 3.3.1. This was done for 10 iterations, and the average results of these iterations was used.

3.5.2 Unseen Test Set

While cross-validation provides a fairly accurate means to analyse how well a classifier will perform, it was also necessary to test the classifier using data from a completely new and unseen source. This would provide an indication as to how well the classifier is able to perform for unseen, real world data. To perform this test, data from the current NRF database was gathered. This data was first manually classified into the 'correct' Dublin Core metadata fields, and then the percentage of correctly classified instances was calculated for each algorithm. The dataset consisted of a total of 109 993 records, the distribution of which can be found in Table 1.

4. PERFOMANCE OF MACHINE LEARNING ALGORITHMS

This section details the results and performance of the various machine learning algorithms that were evaluated.

4.1 Cross-Validation Results

The metric used to assess the performance of the various algorithms was to compare the average percentage of correctly classified instances, based on the 10 fold, 10 iteration cross-validation as discussed in section 3.5.1. The sorted average percentage of correctly classified instances, along with their corresponding standard deviations, are shown in Table 2.

Weka allows the user to choose a 'base algorithm' to which the other algorithms can be statistically compared. Due its popularity, J48 was chosen as the base algorithm to compare the other four algorithms against. The comparisons were done using a two-tailed, paired t-test with a significance level of 5%.

Table 2: Cross-validation results

Algorithm	Percentage Correct	Standard Deviation
Random Forest +	94.28	0.38
J48 (C4.5 decision tree)	93.65	0.37
Logistic Regression -	79.04	0.62
Artificial Neural Network -	76.59	0.91
Naïve Bayes -	54.92	0.71

In Table 2, a plus sign indicates that an algorithm had statistically better performance compared to J48, and a negative sign indicates statistically worse performance.

From the results of this experiment, it appears that Random Forest and J48 perform significantly better than the other chosen algorithms in both accuracy and variance.

4.2 Unseen Data Classification Results

After the cross-validation was performed, the trained algorithms were tested using unseen data as described in section 3.5.2. The results of this evaluation are provided in Table 3 and are sorted by the percentage of correctly classified instances.

While Random Forest performed better than J48 in the cross-validation experiment, it classified close to 10% fewer items correctly on the unseen test set. The two tree based algorithms still outperformed the other algorithms.

Table 3: Unseen NRF data set results

Algorithm	Percentage Correct
J48 (C4.5 decision tree)	79.54
Random Forest	68.08
Logistic Regression	57.69
Artificial Neural Network	55.59
Naïve Bayes	33.91

A confusion matrix was generated for the unseen data test. This matrix is useful in showing how items were incorrectly classified and is shown in Figure 2. The shaded cells along the diagonal represent the count of fields that were correctly classified, and hence we would expect this to be the greatest value in a given row. Some rows contain no values, indicating that this Dublin Core field was not present in the test data set.

The incorrectly classified instances can often be explained by similarities between the correct and incorrect metadata classification field. For example, the majority of 'contributors' were incorrectly classified as 'creators', which could easily be explained by looking at the training data. These fields both contain person names, however 'creator' was far more ubiquitous across the training data sets. 'Description' was often incorrectly classified as 'subject', and 'subject' as 'title'. This is to be expected as these fields often have very similar characteristics in terms of features. Both 'subject' and 'description' were often incorrectly classified as 'language'. This is likely as a result of there not being a sufficient feature to distinguish languages. Such a feature could be added, for example by seeing if a word appears in a list of languages and common language abbreviations.

5. SYSTEM DEVELOPMENT, IMPLEMENTATION AND UI DESIGN

While this project aimed to take a fairly experimental approach to development and analysis, the system being developed needed to provide a front-end, Web-based, UI to access the features so that organisations such as the NRF could use it. As such, it was important to ensure that the system was developed to be of suitable quality for a production release.

5.1 Requirements Gathering

In order to ensure that this project meets the needs of the NRF and the community, it was necessary to gather information on the requirements and expectations of the project, as well as the use cases and need for such software. This was conducted through a variety of means.

5.1.1 Communication with the NRF

E-mail was used as the primary means of communicating with the NRF. Various questions about the software requirements and use cases were asked and answered via e-mail and this proved to be a useful and efficient means to help determine requirements. During the inception of this project, we travelled to the NRF in Johannesburg so that we could view the current system and its issues, as well as investigate the use cases of the NRF. The information gathered through these communications were vital to the scoping and development of this project. These communications highlighted the need for a usable and effective interface, as well as the current problems faced at the NRF (inconsistent metadata, outdated technologies, and difficulties in managing and providing the information to those who need it).

5.1.2 Initial Requirements Survey

From the onset of this project, we wanted to produce software that would not only serve the needs of the NRF, but also the institutional repository community as a whole. Hence it was necessary to seek advice and input from the current DSpace community. This was achieved through the distribution of a survey to a number of DSpace mailing lists. This project was also presented at a satellite meeting (Transforming Libraries with Open Digital Technologies[18]) of the 81st IFLA World Library and Information Conference[19], where input was gathered from the attendants, and a link to the survey was distributed. The survey had 13 respondents and the raw results of the survey can be found in the appendix. The notable results of this survey are that 7 out of the 13 respondents indicated that they have difficulty mapping legacy fields to the appropriate DSpace Dublin Core metadata fields, as well as choosing the appropriate metadata mapping, when importing data. These were the two most prominent issues faced by repository users and hence this software would be well suited to them.

5.1.3 Communication with Project Supervisor

Throughout the course of this project, the project supervisor was consulted through e-mails and during regular project meetings. This proved useful in ensuring the project progressed in an efficient and effective manner, and was being developed to meet the requirements of the community. All meetings were documented in a notebook.

[18] http://conferences.sun.ac.za/ifla-it-2015

[19] http://conference.ifla.org/ifla81

Classified as →	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
A = contributor	0	0	1777	0	3	0	0	0	0	0	0	0	16	0	0
B = coverage	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C = creator	12	0	12639	0	8	1	0	1	45	0	48	2	619	0	5
D = date	0	0	0	16667	0	0	0	0	0	0	0	0	0	0	0
E = description	1	0	1760	0	16295	3	0	2607	21	0	100	45	8678	8	6
F = format	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G = identifier	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H = language	0	0	0	0	0	0	0	13137	0	0	1	0	241	0	2
I = publisher	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J = relation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K = rights	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L = source	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M = subject	1	0	232	0	155	14	6	1049	32	0	2	16	17002	3170	50
N = title	2	67	108	0	123	3	10	1	135	10	77	4	1226	11745	5
O = type	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2: Confusion matrix for NRF data set where highlighted cells indicate correctly classified fields

5.2 Development Framework and Methodology

Throughout this project, the currently accepted software development methodologies and frameworks were used. This section covers some of the methodologies and frameworks used during development.

5.2.1 Programming Language and Framework

Development of the metadata mapper was done using Java Servlets as the development platform. The application provides a front-end user interface through Java Server Pages (JSPs). In order to keep the interface uniform, the core Bootstrap theme[20] (which is used by DSpace) was used. JavaScript, along with JSTL[21], was also used to facilitate the loading of dynamic content and for communication with the server. Java was chosen as the development language as it would allow for easy integration with DSpace, which is Java based. Furthermore, Java is well supported and provides a vast number of external libraries that can be leveraged. As this project is developed using Java Servlets, it can easily be set up to run on any compatible server by deploying the WAR file, and is hence highly portable. Maven[22] was used as a build manager for the project and Bitbucket[23] was used for version control and bug tracking.

5.2.2 Java Libraries

Where possible, open source Java libraries were used, rather than having to re-implement functionality. The two notable libraries used were Weka[24] (for machine learning), and OpenCSV[25] (for parsing the CSV input files).

5.2.3 Integration with DSpace

The metadata mapper was successfully integrated with Darryl Meyer's complementary project that aimed to facilitate batch ingestion into DSpace. Through this ingestion manager, the records that were mapped using the metadata mapper could be uploaded into a DSpace repository.

5.3 Software Development Methodology

In the development of this project, an iterative and Agile software development approach was used. This is discussed in more detail in the following sections.

5.3.1 Agile Development Methodology

There is widespread acceptance among the software development community that an Agile development approach tends to be highly effective for projects where it is likely that the scope and requirements would change as the project progresses. It was expected that as additional feedback and use cases were received, whether through the NRF or other sources, the scope of this project would change. The Agile methodology allowed for easy adaption to changing requirements and ensured that the software being developed would meet the requirements.

5.3.2 Iterative Development and User Centred Design

In addition to ensuring that there was agility in development, the software was produced through an iterative design process. Here, the product was developed through a continuous process of iteratively building onto the software. There were a number of distinct iterations. The first iteration involved a feasibility demonstration that aimed to show the feasibility of the project functionality before major development and a potential design was sketched out. A paper prototype was then conducted with expert users. The aim of this paper prototype was to gather feedback on a potential user interface (UI). Thereafter, an updated UI was developed based on the results of the paper prototype, along with backend functionality. This was then tested through user evaluation, which focused on the design of the interface. The results of this evaluation were considered and incorporated and involved a few minor design changes. Thereafter, minor changes to the design and functionality were implemented as needed.

5.3.2.1 Initial Paper Prototype

A paper prototype was conducted with four postgraduate computer science students in order to gather early feedback from users on the design and functionality of the system. An initial design was drawn up, and feedback and input were requested on this design. The results of this feedback, as well as the initial design sketch, are provided in the appendix. While there were no major changes requested, concerns were raised over the wording used in various areas, as well as the usability of the current design. It was suggested

[20] http://getbootstrap.com/examples/theme/
[21] JSP Standard Tag Library - https://jstl.java.net/
[22] https://maven.apache.org/
[23] https://bitbucket.org/
[24] See section 3.2.
[25] http://opencsv.sourceforge.net/

that various sections be explained in more detail, and that tooltips be used to assists users.

5.3.2.2 *User Evaluation*

Once the design was implemented and a working prototype was developed, five postgraduate computer science students were used to evaluate the software and give feedback. This evaluation was conducted by first providing the users with some background information about DSpace and the tools being developed, and then requesting that they perform certain tasks (with assistance if needed). Problem areas were then identified based on how users used the system, as well as their feedback. The results of this evaluation were positive, however some minor design changes were requested in order to improve the usability of the system.

5.3.2.3 *Usability and User Acceptance Testing*

Once the software was finalised, the system was evaluated through usability testing and the NRF was asked to evaluate whether or not the software had met their initial requirements. This is discussed in more detail in section 6.

5.3.3 *Testing, Documentation and Maintainability*

In order to ensure that the software produced was ready for release into a production environment, it was necessary to ensure that it was thoroughly tested and that it is easy to maintain should it be released. Testing was conducted continuously, whereby each new software iteration would involve basic unit testing by simple print statements and debugging.

The code was thoroughly documented to allow for it to be easily maintained, should it be released in an open source environment, or should support for the product continue after it is released. The documentation for the code is provided in the appendix. Furthermore, the code was developed with the intention of it being easy to modify and upgrade in the future.

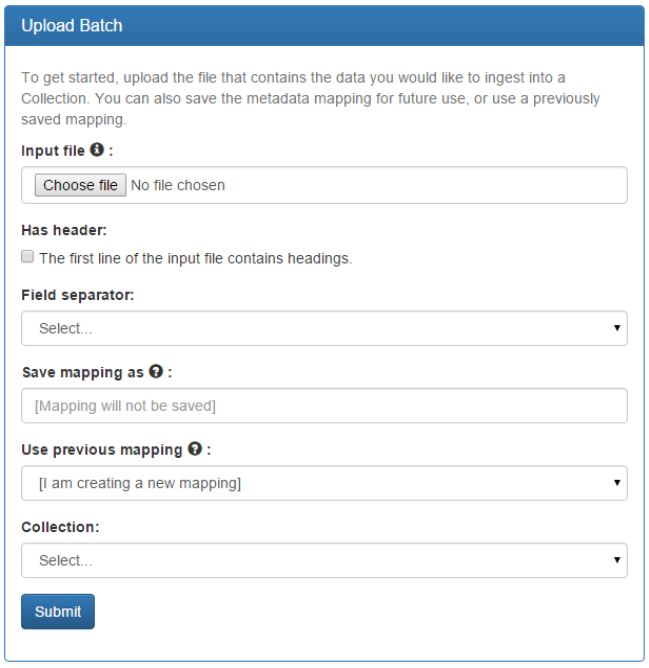

Figure 3: UI of submission page for CSV file.

5.4 Algorithms and Data Structures

This section provides a brief and very high level overview of some of the main algorithms and data structures used.

5.4.1 *Classification of Inputs*

In order to classify new inputs, Weka produces a model based on the trained machine learning algorithm which can then be used. When the user uploads a CSV input file via the interface shown in Figure 3, this file is parsed by iteratively classifying each entry. The entries are fed into the model and classified. A hash map is then used to keep track of the classifications for each field/column. This allows for a score to be shown for each field. For example, if 70% of the entries for a particular field were classified as 'title' and 30% as 'description', we can rank the predictions accordingly. Thereafter, the Dublin Core fields are shown in alphabetical order.

Figure 4 shows a page that contains the results of an automatic metadata mapping. Here the algorithm was able to correctly classify all three fields, with the user only having to specify the secondary field of the Dublin Core 'date' field. Here, 84% of the entries in the field 'Paper title' were correctly classified as 'title'.

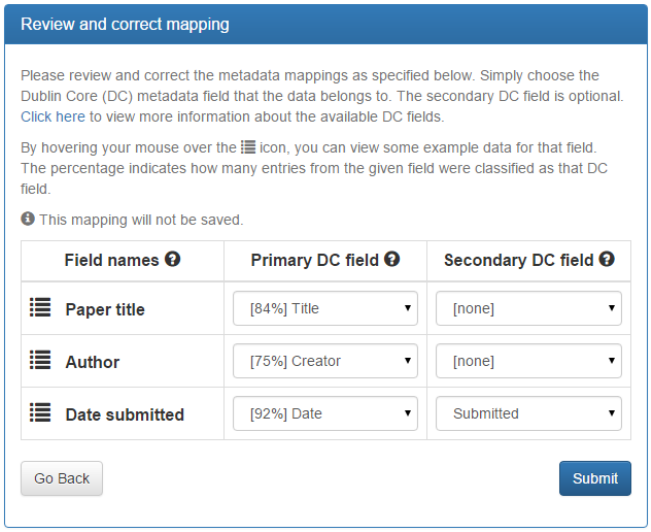

Figure 4: Results page that allows the user to review and correct a metadata mapping.

As the input file is parsed iteratively, there should be no issues with processing very large input files and the program has successfully parsed input files with over 10 000 records.

5.4.2 *Saving and Loading Mappings*

A typical use case of the NRF is that an institution will submit data to them on a regular basis, which would need to be uploaded into DSpace. It is a fairly safe assumption that data from the same institution will likely use the same CSV format and hence the metadata mappings can be reused. This is achieved by allowing the user to save a mapping into a PostgreSQL[26] database running on the server. The user can save the mapping as a custom name, and then load this mapping in the future. This stored mapping can then be applied to the new data, negating the need for the user to have to manually check and update the mappings.

[26] http://www.postgresql.org/

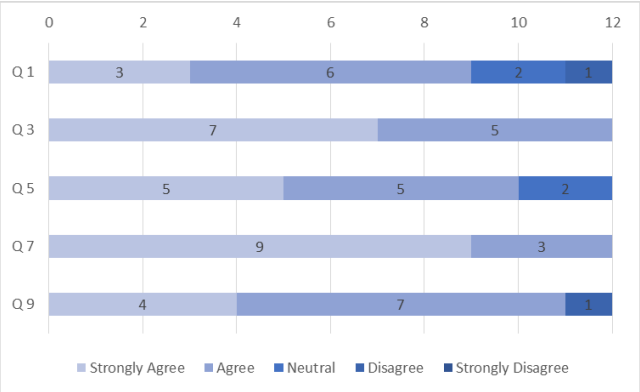

Figure 5: Cumulative distribution of SUS results for positively phrased questions.

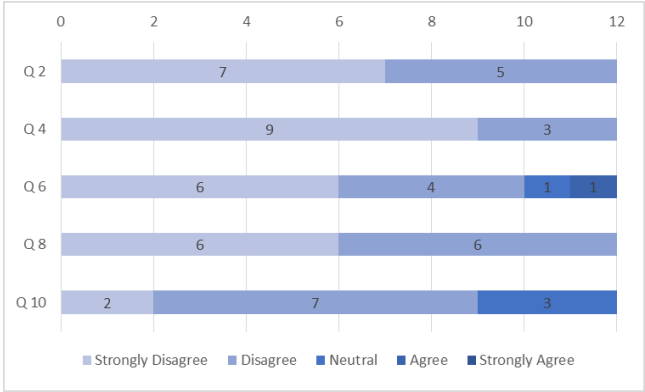

Figure 6: Cumulative distribution of SUS results for negatively phrased questions.

6. SOFTWARE USABILITY AND ACCEPTANCE

In this section, the results of usability testing, as well as the feedback received on the final system, are discussed.

6.1 System Usability Testing

As the system being developed required a front-end design to interact with the system, it was imperative that the UI was easy to use and intuitive. In order to test the tool's usability, a standard usability test was used, the System Usability Scale (SUS) [7]. The test was conducted on a near final version of the software and the users were requested to complete a set of tasks based on a template provided by Snyder [32]. SUS consists of 10 questions, where responses are constrained to a Likert scale that ranges from 'strongly disagree' to 'strongly agree'. The raw data from the usability test on 12 participants, as well as the mode response for each question is provided in the appendix. The 10 questions alternate between positively and negatively phrased questions and hence the results are divided amongst Figure 5 and Figure 6. Each figure shows the cumulative distribution of the responses for each question. Note that the colour scale has been inverted in Figure 6 so that light shading indicates a positive response and vice versa. The 10 SUS questions are given below:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

SUS was developed to try to represent the overall usability of a system through a single number, ranging from 0 to 100, with 100 being a 'perfect' score. The raw data, as well as the mode responses, do not appear to indicate any particular usability issues. The overall SUS score achieved was just above 84. While it may be considered ineffective to render usability down to a single metric, there has been much research on SUS and interpreting scores and their accuracy. Bangor et al. [2] added an 11th question to nearly 1000 SUS surveys that required respondents to rank the system on a seven-point adjective-anchored Likert scale. They achieved a statistically significant and strong correlation between SUS scores and the adjective based scale. The authors also map a SUS score onto a letter grade scale and into acceptability ranges as shown in Figure 7.

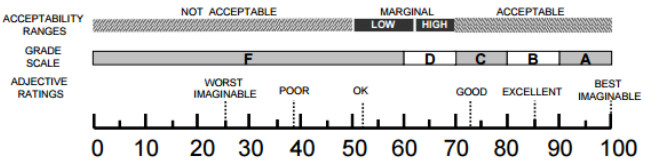

Figure 7: Interpreting SUS scores by comparing them to an adjective-based scale [2].

Based on this scale, the usability of the product developed can be classified as 'excellent' and would be of an acceptable standard for a production environment.

6.2 General Feedback

Through the course of the development of this project, feedback was collected and gathered from a variety of sources. The overall feedback on this project was positive, with all criticism being constructive and leading to consistent improvements and updates to the design of the product. General comments from the participants were positive, with praise being given to the final UI theme and design. Furthermore, participants also indicated that they felt that the use of machine learning in this tool added an element of 'excitement' to using the tool. From comments left by users of the final system, it was clear that the intial issues of poor wording and usability were effectively adressed, as no concerns were raised about wording and users were able to easily navigate the software without assistance.

6.3 Acceptance of the Tool by the NRF

User acceptance testing involved investigating whether or not the initial requirements, as provided by the NRF, were met by the tools that were produced through this project. In order to determine whether or not the tools produced fulfilled the use cases of the NRF, a questionnaire was prepared for the NRF, and the tools were made accessible by hosting them on a Web server. Instructions were then provided (see appendix) to assist the NRF in completing some tasks that would help showcase the metadata mapper. The NRF was also asked to complete a SUS questionnaire. The raw results of this are provided in the appendix.

The NRF indicated that the metadata mapper met their requirements and that they were pleased with the results. They did however indicate that it would be useful if custom Dublin Core fields could be used, instead of being limited to the standard DSpace Dublin Core fields. The NRF usability survey results were positive, with the only concern being that they felt they had to learn a lot of things before using this system. The overall NRF SUS score achieved was 90.

7. ETHICAL, PROFESSIONAL, AND LEGAL ISSUES

As this tool will be used to migrate and import data, it was important to ensure that data is not intentionally, or unintentionally, modified, deleted or added to the repository. Furthermore, the outputs of this project may be released to the community and, as such, there was a professional responsibility to ensure that the output is of a high standard.

All user testing was conducted through simple surveys and usability testing, which did not raise any ethical issues.

8. CONCLUSIONS

In this section, some of the conclusions that have emerged through the development and outcomes of this project are discussed.

8.1 The Metadata Mapper is Effective but can be Improved

This project aimed to develop a tool that would help to transition DSpace into a RIMS. It was also important to be mindful of the requirements and use cases of the NRF. From the experimental results, and usability and user acceptance tests, it was clear that this tool did satisfy the initial aims of the project. The user interface was widely accepted and the experimental results indicated that there were algorithms that were capable of producing accurate classifications that would help make the metadata mapping process simpler and quicker.

The NRF did however indicate that the tool currently lacks the ability to use custom metadata fields. Furthermore, while every effort was taken to ensure that the tool developed was of suitable quality for a production environment, it would be ideal if additional testing could take place to ensure that the tool is as robust and bug free as possible. Another feature which is lacking is the ability to remove saved mappings from the database through the Web-based interface. These issues are addressed in section 9.

8.2 Decision Trees are Highly Effective at Classifying Textual Based Data

From the results of the experiments, it was interesting to note that both tree based algorithms performed considerably better than the other algorithms. It is clear that they provide an effective means of classifying textual data. Furthermore, these algorithms were amongst the quickest to train and test on, and proved to be highly suitable candidates to be used in this software.

8.3 This Tool Could be Integrated with the NRF's DSpace Repository

Should only the default DSpace Dublin Core fields be used, this tool is currently of suitable quality to be used by the NRF as a means of adding data to the DSpace repository on a continuous basis. It would be ideal if support for the metadata mapper continues so that any bugs identified by the NRF could be patched. Furthermore, the NRF may become aware of certain features that would make a good addition to this tool. These features could then be incorporated and may prove useful to other organisations that would be interested in the automatic and manual metadata mapper.

8.4 The Metadata Mapper is Easy to Use and Effective

The results of the usability testing, as well as the user acceptance testing, indicate that the outcome of this project is an easy to use tool, with an effective and usable UI that provides a good user-experience. Through testing and comparison to the current system of performing batch ingestion into DSpace, it is clear that this tool provides an effective means to add data into a DSpace repository and can prove to be a useful tool for helping to migrate data from a legacy system into an institutional repository.

8.5 The Metadata Mapper Helps Transform DSpace into a RIMS

While this tool alone cannot constitute the transformation of DSpace into a RIMS, it can prove to be a useful addition to DSpace for institutions (such as the NRF) that act as an aggregator of research products for a variety of institutions. This is achieved through the ability to easily map metadata fields and save and reuse these mappings.

9. FUTURE WORK

In this section, some of the potential areas for future work in this field and on this tool are discussed. Due to the limited time available to complete this project, there were certain features that were purposefully not included. Furthermore, as this project progressed, certain areas in which future work could focus became apparent.

An interesting future study would be to perform a feature analysis whereby various additional features could be incorporated as inputs into various machine learning algorithms. This study could then investigate which feature set performs best and whether or not it is able to provide better classification results than what has currently been achieved. It would also be interesting to evaluate other potential machine learning algorithms, as well as to tweak and optimise the parameters used in the algorithms discussed in this paper. A future expansion of this project could allow for the user to specify custom metadata fields. The user should also be able to delete, view and modify saved mappings, which is currently not possible.

Should the above improvements be implemented, it would further encourage institutions to adopt this tool as a means of adding RIMS features to DSpace and in so doing, benefit the research management community.

10. ACKNOWLEDGEMENTS

I would like to thank my project partner, Darryl Meyer, for his commitment and help, as well as Lazarus Matizirofa (from the NRF) for providing valuable input and feedback throughout the course of this project. Finally, my sincere thanks and appreciation are extended to my project supervisor, Associate Professor Hussein Suleman, for his constant guidance, support, encouragement and sound advice.

11. REFERENCES

[1] W. Y. Arms. 1995. Key Concepts in the Architecture of the Digital Library. *D-lib Magazine*, 1, 1.

[2] A. Bangor, P. Kortum and J. Miller. 2009. Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4, 3, 114-123.

[3] J. Berlin and A. Motro. 2002. Database schema matching using machine learning with feature selection. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE 2002)*, Toronto, Canada, Springer, 452-466.

[4] J. Bisbal, D. Lawless, B. Wu, J. Grimson, V. Wade, R. Richardson and D. O'Sullivan. 1997. An overview of legacy information system migration. In *Proceedings of the Fourth Asia-Pacific Software Engineering and International Computer Science Conference (APSEC '97 / ICSC '97)*, Clear Water Bay, Hong Kong, IEEE Computer Society, Washington, DC, USA, 529-530.

[5] L. Breiman. 2001. Random Forests. *Machine Learning*, 45, 1, 5-32.

[6] M. L. Brodie and M. Stonebraker. 1995. Migrating legacy systems: gateways, interfaces & the incremental approach. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[7] J. Brooke. 1996. SUS: A "quick and dirty usability" scale. In *Usability evaluation in industry*, I. L. McClelland (Ed.). Taylor and Francis, London, UK, 189-194.

[8] R. Brownlee. 2009. Research data and repository metadata: policy and technical issues at the University of Sydney Library. *Cataloging & Classification Quarterly*, 47, 3/4, 370-379.

[9] A. Doan, P. Domingos and A. Y. Halevy. 2001. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data (SIGMOD '01)*, Santa Barbara, California, USA, ACM, 509-520.

[10] A. Doan, P. Domingos and A. Y. Halevy. 2000. Learning Source Description for Data Integration. In *Proceedings of the Third International Workshop on the Web and Databases*, Dallas, TX, USA, ACM SIGMOD, 81-86.

[11] B. Haslhofer and W. Klas. 2010. A Survey of Techniques for Achieving Metadata Interoperability. *ACM Computing Surveys*, 42, 2.

[12] T. Hastie, R. Tibshirani and J. Friedman. 2005. Random Forests. In *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Anonymous (Ed.). Springer, New York, NY, USA, 587-604.

[13] D. W. Hosmer Jr, S. Lemeshow and R. X. Sturdivant. 2013. Applied Logistic Regression. John Wiley & Sons, Hoboken, NJ, USA.

[14] R. Kahn and R. Wilensky. 2006. A framework for distributed digital object services. *International Journal on Digital Libraries*, 6, 2, 115-123.

[15] R. Kohavi. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on AI (IJCAI-95)*, Montreal, Quebec, Morgan Kaufmann, Los Altos, CA, 1137-1145.

[16] W. Li and C. Clifton. 2000. SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data & Knowledge Engineering*, 33, 1, 49-84.

[17] W. Li, C. Clifton and S. Liu. 2000. Database integration using neural networks: Implementation and experiences. *Knowledge and Information Systems*, 2, 1, 73-96.

[18] W. Li and C. Clifton. 1994. Semantic Integration in Heterogeneous Databases Using Neural Networks. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94)*, Santiago de Chile, Chile, Morgan Kaufmann Publishers Inc., 12-15.

[19] J. Madhavan, P. A. Bernstein and E. Rahm. 2001. Generic schema matching with Cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01)*, Rome, Italy, Morgan Kaufmann, San Francisco, CA, USA, 49-58.

[20] R. J. Miller, L. M. Haas and M. A. Hernández. 2000. Schema Mapping as Query Discovery. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB '00)*, Cairo, Egypt, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 77-88.

[21] R. Mishra, S. Vijayanand, P. P. Noufal and G. Shukla. 2007. Development of ETD Repository at IITK Library using DSpace. In *International Conference on Semantic Web and Digital Libraries (ICSD-2007)*, Bangalore, India, Indian Statistical Institute, 249-259.

[22] T. Mundle. 2007. Digital retrospective conversion of theses and dissertations: an in house project. In *8th International symposium on electronic theses and dissertations (ETD 2005)*, Sydney, Australia, NDLTD.

[23] A. M. Ouksel and A. Sheth. 1999. Semantic interoperability in global information systems. *ACM Sigmod Record*, 28, 1, 5-12.

[24] J. R. Quinlan. 1993. C4. 5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[25] E. Rahm and P. A. Bernstein. 2001. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10, 4, 334-350.

[26] T. Ribaric. 2009. Automatic Preparation of ETD Material from the Internet Archive for the DSpace Repository Platform. *Code4Lib Journal* 8.

[27] S. Russell and P. Norvig. 2009. Artificial Intelligence: A Modern Approach. Prentice Hall Press, Upper Saddle River, NJ, USA.

[28] F. Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34, 1, 1-47.

[29] A. P. Sheth and J. A. Larson. 1990. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys (CSUR)*, 22, 3, 183-236.

[30] P. Shvaiko and J. Euzenat. 2005. A survey of schema-based matching approaches. In *Journal on Data Semantics IV*, Anonymous (Ed.). Springer, 146-171.

[31] M. Smith, M. Barton, M. Bass, M. Branschofsky, G. McClellan, D. Stuve, R. Tansley and J. H. Walker. 2003. DSpace: An open source dynamic digital repository. *D-Lib Magazine*, 9, 1.

[32] C. Snyder. 2003. Paper prototyping: The fast and easy way to design and refine user interfaces. Morgan Kaufmann, San Francisco, CA.

[33] R. Tansley, M. Bass, D. Stuve, M. Branschofsky, D. Chudnov, G. McClellan and M. Smith. 2003. The DSpace institutional digital repository system: current functionality. In *Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries (JCDL'03)*, Houston, TX, USA, IEEE Computer Society, Washington, DC, USA, 87-97.

[34] K. M. Ting and I. H. Witten. 1999. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10, 271-289.

[35] P. R. S. Visser, D. M. Jones, T. J. M. Bench-Capon and M. J. R. Shave. 1997. An analysis of ontology mismatches; heterogeneity versus interoperability. In *AAAI 1997 Spring Symposium on Ontological Engineering*, Stanford CA., USA, AAAI, 164-172.

[36] H. Wache. 2003. Semantische mediation für heterogene informationsquellen. *KI*, 17, 4, 56.

[37] M. P. Walsh. 2010. Batch Loading Collections into DSpace: Using Perl Scripts for Automation and Quality Control. *Information Technology and Libraries*, 29, 3, 117-127.

[38] I. H. Witten, D. Bainbridge, R. Tansley, C. Huang, K. Don and N. Z. Hamilton. 2005. A Bridge between Greenstone and DSpace. *D-Lib Magazine*, 11, 9.

[39] D. H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5, 2, 241-259.

[40] B. Wu, D. Lawless, J. Bisbal, J. Grimson, V. Wade, D. O'Sullivan and R. Richardson. 1997. Legacy system migration: A legacy data migration engine. In *Proceedings of the 17th International Database Conference (DATASEM'97),* Brno, Czech Republic, 129-138.

[41] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu and S. Y. Philip. 2008. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14, 1, 1-37.