Information Services & Use 33 (2013) 77–86
DOI 10.3233/ISU-130695
IOS Press

Deduplication of metadata harvested from Open Archives Initiative repositories

Piotr Wendykier
Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw, ul. Prosta 69, 00-838 Warsaw, Poland
E-mail: p.wendykier@icm.edu.pl

Abstract. Open access (OA) is a way of providing unrestricted access via the Internet to peer-reviewed journal articles as well as theses, monographs and book chapters. Many open access repositories have been created in the last decade. There is also a number of registry websites that index these repositories. This article analyzes the repositories indexed by the Open Archives Initiative (OAI) organization in terms of record duplication. Based on the sample of 958 metadata files containing records modified in 2012 we provide an estimate on the number of duplicates in the entire collection of repositories indexed by OAI. In addition, this work describes several open source tools that form a generic workflow suitable for deduplication of bibliographic records.

Keywords: Deduplication, record linkage, metadata, Dublin core, open access, OAI–PMH

1. Open Archives Initiative

The Budapest Open Access Initiative [1] defines open access documents as follows: "By open access, we mean its immediate, free availability on the public internet, permitting any users to read, download, copy, distribute, print, search or link to the full text of these articles, crawl them for indexing, pass them as data to software or use them for any other lawful purpose". The number of open access repositories has grown significantly in the last decade. As a result, different organizations started creating registry websites that index these repositories. One of them is maintained by the Open Archives Initiative (OAI) – an association that develops and promotes interoperability standards that aim to facilitate the efficient dissemination of content. The main goal of OAI is to enhance access to all kinds of e-print archives independent of the type of content offered as well as the economic mechanisms surrounding that content. Current OAI projects include the Open Archives Initiative Protocol for Metadata Harvesting (OAI–PMH) and Open Archives Initiative Object Reuse and Exchange (OAI–ORE). The latter one defines standards for the description and exchange of aggregations of Web resources. These aggregations may combine distributed resources with multiple media types including text, images, data and video. In this article we focus only on the bibliographic records that can be acquired by using OAI–PMH protocol.

1.1. OAI–PMH

The OAI–PMH [14] is used to collect the metadata descriptions of the records in an archive so that services can be built using metadata from many archives. It uses XML over HTTP and the current

version 2.0 was updated in 2008. OAI–PMH is based on a client–server architecture, in which *harvesters* request information on updated records from repositories. Requests for data can be based on a date-stamp range, and can be restricted to named sets defined by the provider. Data providers are required to provide XML metadata in a Dublin Core format [19], and may also provide it in other XML formats. The specification of OAI–PMH defines an *item* as a container that stores or dynamically generates metadata about a single resource in multiple formats, each of which can be harvested as records via the protocol. Each item has an *identifier* that is unique within the scope of the repository. The format of the unique identifier must correspond to that of the URI (Uniform Resource Identifier) syntax. A *record* is metadata expressed in a single format and is returned in an XML-encoded byte stream in response to an OAI–PMH request for metadata from an item. The XML-encoding of records is organized into three parts: *header* (contains the unique identifier of the item and properties necessary for selective harvesting), *metadata* (a single manifestation of the metadata from an item), and *about* (an optional and repeatable container to hold data about the metadata part of the record). An example of the OAI–PMH record with metadata expressed in the Dublin Core format is shown in Listing 1.

```
<header>
  <identifier>oai:arXiv:cs/0112017</identifier>
  <datestamp>2002-02-28</datestamp>
  <setSpec>cs</setSpec>
  <setSpec>math</setSpec>
</header>
<metadata>
 <oai_dc:dc xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
            xmlns:dc="http://purl.org/dc/elements/1.1/"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/
            http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
   <dc:title>Using Structural Metadata to Localize Experience of Digital Content</dc:title>
   <dc:creator>Dushay, Naomi</dc:creator>
   <dc:subject>Digital Libraries</dc:subject>
   <dc:description>With the increasing technical sophistication of both information consumers
    and providers, there is increasing demand for more meaningful experiences of digital
    information. We present a framework that separates digital object experience, or rendering,
    from digital object storage and manipulation, so the rendering can be tailored to particular
    communities of users.
   </dc:description>
   <dc:description>Comment: 23 pages including 2 appendices, 8 figures</dc:description>
   <dc:date>2001-12-14</dc:date>
   <dc:type>e-print</dc:type>
   <dc:identifier>http://arXiv.org/abs/cs/0112017</dc:identifier>
 </oai_dc:dc>
</metadata>
<about>
  <provenance
     xmlns="http://www.openarchives.org/OAI/2.0/provenance"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/provenance
     http://www.openarchives.org/OAI/2.0/provenance.xsd">
    <originDescription harvestDate="2002-02-02T14:10:02Z" altered="true">
      <baseURL>http://the.oa.org</baseURL>
      <identifier>oai:r2:klik001</identifier>
      <datestamp>2002-01-01</datestamp>
      <metadataNamespace>http://www.openarchives.org/OAI/2.0/oai_dc/</metadataNamespace>
    </originDescription>
  </provenance>
</about>
```

Listing 1. OAI–PMH record with metadata expressed in the Dublin Core format. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/ISU-130695.)

OAI–PMH supports the following six types of requests (aka. verbs) [14]:

- *Identify* – retrieves information about a repository such as *repositoryName*, *baseURL*, *protocolVersion*, *earliestDatestamp*, *deletedRecord* and *granularity*.
- *ListMetadataFormats* – retrieves the metadata formats available from a repository. Possible formats include oai_dc, olac and perseus.
- *ListSets* – retrieves the set structure of a repository, which is useful for selective harvesting.
- *GetRecord* – retrieves an individual metadata record from a repository. It is required to specify the identifier of the item from which the record is requested and the format of the metadata that should be included in the record.
- *ListRecords* – is used to harvest records from a repository that match specified criteria such as lower and upper bounds for the datestamp.
- *ListIdentifiers* – is an abbreviated form of ListRecords, retrieving only headers rather than records.

Each type of request takes additional arguments. For ListRecords, the following arguments are supported:

- from – an optional argument which specifies a lower bound for datestamp-based selective harvesting;
- until – an optional argument which specifies a upper bound for datestamp-based selective harvesting;
- set – an optional argument which specifies set criteria for selective harvesting;
- resumptionToken – an exclusive argument with a value that is the flow control token returned by a previous ListRecords request that issued an incomplete list;
- metadataPrefix – a required argument (unless the exclusive argument resumptionToken is used) that specifies the metadataPrefix of the format that should be included in the metadata part of the returned records.

For example, the request in the form http://arXiv.org/oai2?verb=GetRecord&identifier=oai:arXiv.org:cs/0112017&metadataPrefix=oai_dc will return a record with identifier oai:arXiv.org:cs/0112017 in the Dublin Core format (metadataPrefix=oai_dc).

1.2. Dublin Core

The Dublin Core metadata terms [19] are a set of vocabulary terms which can be used to describe resources for the purposes of discovery. The terms can be used to describe a full range of web resources: video, images, web pages etc. and physical resources such as books and objects like artworks. The Simple Dublin Core Metadata Element Set (DCMES) [5] consists of 15 metadata elements: title, creator, subject, description, publisher, contributor, date, type, format, identifier, source, language, relation, coverage and rights. Each Dublin Core element is optional and may be repeated. The Dublin Core Metadata Initiative (DCMI), a public, not-for-profit company, has established standard ways to refine elements and encourage the use of encoding and vocabulary schemes. There is no prescribed order in Dublin Core for presenting or using the elements. The Dublin Core became ISO 15836 standard in 2006 and is a required metadata format in OAI–PMH protocol. Listing 1 shows how the Dublin Core format is used in the OAI–PMH record.

1.3. Repositories

The OAI–PMH has become widely adopted by many digital libraries, institutional repositories, and digital archives. There are several large registries of OAI-compliant repositories:

(1) The Open Archives list of registered OAI repositories [16].
(2) Openarchives.eu: the European Guide to OAI–PMH compliant repositories in the world [17].
(3) ScientificCommons.org: a worldwide service and registry [20].

This work present results only from the first registry. As of March 6, 2013, there are 1934 repositories registered in the Open Archives list including ArXiv, PubMed Central, Nature Publishing Group Metadata Repository, NASA Technical Reports Server and CERN Document Server. Since Open Archives list stores only links to individual repositories, it is not possible to query aggregated results from multiple collections. To enable such functionality, one would have to build a digital library containing metadata from all repositories. However, to ensure uniqueness of each record in the resulting library, a deduplication operation needs to be performed. The rest of this paper describes the problem of deduplication in practice, where the database of over four million bibliographic records is analyzed.

2. Record linkage and deduplication

Databases play very important role in today's world. Many industries depend on the accuracy of data stored in databases to carry out operations. Therefore, for any analysis of database resources, the uniqueness of every record is very desirable. Unfortunately, due to the lack of unique global identifier in each table, the process of linking of two or more tables may introduce duplicates. This section, after formal definition of the problem, describes the main features of a particular software package for record linkage and deduplication.

2.1. Definition

For a given sets A and B of records, a *record linkage* is a process of finding a partition of $A \times B$ consisting of sets M (matched), U (unmatched) and P (possibly matched) that satisfy $M = \{(a, b)|a = b\}$ and $U = \{(a, b)|a \neq b\}$. In a classical probabilistic approach to record linkage defined by Fellegi [7], a vector of similarity scores (or agreement values) is first computed for each pair. Then, the pair is classified as either a match or non-match (or possibly matched) based on an aggregate of the similarity scores. Classification methods have been widely studied [6,8,21] and the current state-of-the-art algorithms include rule-based methods that allow human experts to specify matching rules, unsupervised learning methods such as Expectation–Maximization (EM) that learns the weights or thresholds without relying on labeled data, and supervised learning methods that use labeled data to train a model, such as a decision tree, a naïve Bayesian or the Support Vector Machine (SVM). For computing similarities, various distance functions are used and studied [4,6,13,18].

Data *deduplication* refers to the process of identifying duplicates in a single set of records. Therefore, all the classification methods and distance functions defined for the record linkage, apply also to the problem of deduplication.

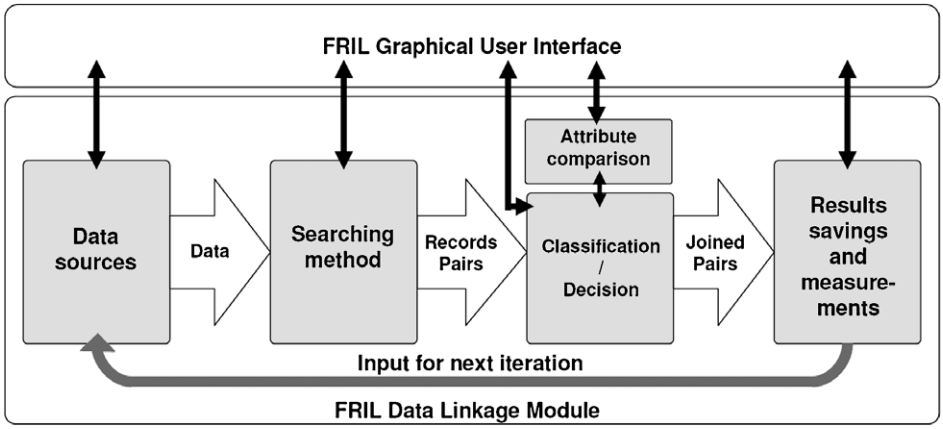

Fig. 1. A general architecture of FRIL (taken from [12]).

2.2. *Fine-grained Record Integration and Linkage Tool*

There are several open source tools for record linkage and deduplication with Link Plus [3] and Link King [2] being the most widely known examples. None of them, however, offers as many features and configuration settings as Fine-grained Record Integration and Linkage Tool (FRIL) [11,12]. FRIL (current version 2.1.5) is an open source Java software that incorporates a collection of record distance metrics, search methods, and analysis tools. In addition, FRIL provides a rich set of user-tunable parameters to enhance linking performance and accuracy [10]. FRIL's data reconciliation feature allows merging two attributes in data source into one attribute (e.g. merging first and last names stored separately into one attribute), splitting given attribute into few columns (e.g. splitting name attribute into two attributes, first and last names) as well as trimming/replacing values in a given attribute via regular expressions. A general architecture of FRIL is shown in Fig. 1.

2.2.1. *Data sources*

Data sources visible in Fig. 1 provide input data for FRIL. The following data sources are supported:

- CSV file – text file where data is separated by comma, semicolon or other separator character.
- Excel file – file saved in the Microsoft Excel format.
- Fixed width columns text file – each row in that file is a separate record, and every attribute should have fixed length.
- Database – the data is pulled from a database. Currently supported: Microsoft SQL Server, PostgreSQL, MySQL and Oracle.

Surprisingly, the only format that is supported by a result saver is a CSV file with ASCII encoding.

2.2.2. *Search and blocking methods*

Search methods, defined in the context of record linkage, refer to algorithms for determining which pairs of records to compare between the data sources A and B and the attributes on which comparisons are made. FRIL implements three types of search methods: a *nested loop join*, a *blocking search* and a *sorted neighborhood* method. The nested loop join performs an all-to-all comparison between A and B and is useful for small data sources. The blocking search method scans all the records in both data sources and divides them into sets called *buckets* based on some predefined blocking conditions. Under the assumption that no matches occur across different blocks, only records that fall into the same bucket

are compared. The sorted neighborhood method first sorts records of A and B over the relevant attributes, and follows by comparing only records within fixed windows A and B of records as these windows are advanced along the input data sets.

In the context of deduplication, the equivalent of a search method is a blocking technique. FRIL implements three blocking algorithms: based on the value of blocking attribute, based on the Soundex code of a blocking attribute and based on the prefix of a blocking attribute.

2.2.3. Distance metrics

Distance metrics are one of the most important components in the process of deduplication. They allow to provide information on how the attributes should be treated (as strings, numbers, dates) and what are allowable discrepancies between values of the attributes. Each distance function maps similarity between values of attributes onto value in the range [0, 1], where 0 means "are not similar in terms of this distance function" and 1 means "are exactly the same in terms of this distance function". FRIL implements seven distance functions:

- *Equal fields boolean distance* – returns 1 if compared values are exactly the same, otherwise it returns 0.
- *Edit distance* – it tests how many operations need to be applied to the first string so that it is converted to the other string. Possible operations include deleting a character, inserting a character or switching two characters that are next to each other. This metric requires two parameters: approve level and disapprove level. The score of edit distance function is calculated using the following formula:

$$d_e(s_1, s_2) = \begin{cases} 0, & e(s_1, s_2) > d \cdot \max\big(|s_1|, |s_2|\big), \\ 1, & e(s_1, s_2) < a \cdot \max\big(|s_1|, |s_2|\big), \\ \dfrac{d \cdot \max(|s_1|, |s_2|) - e(s_1, s_2)}{(d - a) \cdot \max(|s_1|, |s_2|)}, & \text{otherwise,} \end{cases}$$

  where $e(s_1, s_2)$ is the edit distance between two strings, a is an approve level and d is a disapprove level.
- *Jaro–Winkler distance* – the score of this distance is calculated as $d_w(s_1, s_2) = d_j + l \cdot p \cdot (1 - d_j)$, where d_j is the Jaro–Winkler distance for strings s_1 and s_2, l is the length of common prefix at the start of the string up to a maximum of 4 characters and p is a constant scaling factor for how much the resulting distance d_w is adjusted upwards for having common prefixes. The Jaro–Winkler distance is defined as follows:

$$d_j = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m - t}{m}\right),$$

  where m is the number of matching characters, and t is the number of transpositions. Two characters from $s_1$and s_2 are matching, if they are not farther than $\lfloor \frac{\max(|s_1|,|s_2|)}{2} \rfloor - 1$ positions apart. The number of matching (but different sequence order) characters divided by 2 defines the number of transpositions.
- *Q-grams distance* – both input strings are first divided into q-grams (substrings of length q). Next, a count of items that do not appear in the intersection of those two sets is calculated. Configuration of this distance requires specification of q, approve level and disapprove level.

- *Soundex* – implements the Soundex algorithm [9]. Requires Soundex length, approve and disapprove levels.
- *Numeric distance* – this distance allows users to specify a range of values that will have a non-zero match score.
- *Date distance* – input values are treated as dates.

3. Experimental results

The process of data analysis discussed in this work is divided into three steps: data harvesting, database creation and deduplication. The details of each step are presented in this section.

3.1. Testbed

The process of data harvesting was performed on a Linux server equipped with quad-core AMD Opteron 2356 (2.30 GHz) and 16 GB RAM. The machine was running Debian GNU/Linux 6.0.3 and Oracle Java SE 6u26. Database creation and deduplication was performed on a notebook computer equipped with Intel Core i5-2430M CPU (2.40 GHz) and 8 GB RAM. The notebook was running Windows 7 64-bit operating system, MySQL 5.5 and Oracle Java SE 7u4.

3.2. Data harvesting

An XML formatted list of base URLs of registered data providers is available at

http://www.openarchives.org/Register/ListFriends.

To harvest the records from all providers we used an open source Java program called OAIHarvester2 [15]. For the purpose of this article, only records added in 2012 (from = 2012-01-01) was harvested and the Dublin Core was used as a metadata format. This step finished within 12 hours and the harvester was able to download 1679 files out of which 717 was empty (no records were modified in 2012) and 4 files had an invalid format. Remaining 958 nonempty files, containing 4,666,194 records, were used in the next stage.

3.3. Database creation

MySQL, the world's most used open source relational database management system (RDBMS), was chosen to store the records. For the purpose of duplicates identification, the following three terms from Dublin Core metadata was chosen: *title*, *subject* and *identifier*. In addition, the table in the RDBMS contained the *id* (primary key) and the *repository* columns, which were not used in the process of deduplication. The process of populating the database took about an hour and it was the last step needed to prepare the input data for FRIL.

3.4. Deduplication

The process of deduplication was performed by using the latest version of FRIL. It was run three times with different distance metrics: equal fields boolean distance, Jaro–Winkler distance and edit distance. Listing 2 shows FRIL configuration file with Jaro–Winkler distance. In that file, an XML tag

```
<?xml version="1.0" encoding="UTF-8"?><configuration deduplication="true">
  <left-data-source class="cdc.impl.datasource.jdbc.JDBCDataSource" name="sourceA">
    <params>
      <param name="columns-select" value="select * from oai.titles"/>
      <param name="source-name" value="sourceA"/>
      <param name="driver" value="com.mysql.jdbc.Driver"/>
      <param name="table" value="articles"/>
      <param name="password" value="removed"/>
      <param name="user" value="removed"/>
      <param name="url" value="jdbc:mysql://localhost:3306/oai"/>
    </params>
    <row-model>
      <column column="id" converter="cdc.datamodel.converters.DummyConverter" name="id">
        <empty-values/>
      </column>
      <column column="title" converter="cdc.datamodel.converters.DummyConverter" name="title">
        <empty-values/>
      </column>
      <column column="subject" converter="cdc.datamodel.converters.DummyConverter" name="subject">
        <empty-values/>
      </column>
      <column column="identifier" converter="cdc.datamodel.converters.DummyConverter" name="identifier">
        <empty-values/>
      </column>
      <column column="repository" converter="cdc.datamodel.converters.DummyConverter" name="repository">
        <empty-values/>
      </column>
    </row-model>
    <preprocessing>
      <deduplication>
        <deduplication-condition acceptance-level="100">
          <condition class="cdc.impl.distance.JaroWinkler" column="title" weight="34">
            <params>
              <param name="pref-weight" value="0.1"/>
              <param name="pref-length" value="4"/>
            </params>
          </condition>
          <condition class="cdc.impl.distance.JaroWinkler" column="subject" weight="33">
            <params>
              <param name="pref-weight" value="0.1"/>
              <param name="pref-length" value="4"/>
            </params>
          </condition>
          <condition class="cdc.impl.distance.JaroWinkler" column="identifier" weight="33">
            <params>
              <param name="pref-weight" value="0.1"/>
              <param name="pref-length" value="4"/>
            </params>
          </condition>
        </deduplication-condition>
        <hashing-function columns="title,title" hash="soundex(5)"/>
        <minus-file file="duplicates.csv"/>
        <dedupe-file file="deduplicated-source.csv"/>
      </deduplication>
    </preprocessing>
  </left-data-source>
</configuration>
```

Listing 2. FRIL configuration file. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/ISU-130695.)

<params> contains information about database connection, a <row-model> tag lists all the columns that will be stored in the FRIL output files (see tags <minus-file> and <dedupe-file>) and the deduplication settings are stored in a <deduplication> tag. Three columns were selected to compute similarity measure between records and the following weights were assigned to these columns: 34 (*title*), 33 (*subject)* and 33 *(identifier*). For the Jaro–Winkler metric, the prefix length was set to 4

Table 1
Deduplication results

Metric	Number of duplicates	Percentage (%)	Time in hours
Equal fields boolean	318,818	6.83	0.34
Jaro–Winkler	126,721	2.72	13.45
Edit distance	319,099	6.84	64.66

and the scaling factor was equal 0.1. The edit distance was configured with approve level $= 0.0$ and disapprove level $= 0.1$. The duplicate acceptance level was set to 100 for all metrics.

FRIL considers two records as duplicates if the sum of products of distance metric value for compared attribute between the records multiplied by the weight for the attribute is greater or equal to the duplicate acceptance level. FRIL looks for duplicates only among the records that end up in the same block. The Soundex code of the *title* column with a length equal to 5 was chosen as a blocking method.

Table 1 shows the results of deduplication: the number of identified duplicates, the percentage of all records and the time of computations. These values clearly demonstrate that the collection of 4,666,194 entries contains significant amount of duplicates. In addition, the number of duplicates depends considerably on the metric – the equal fields boolean distance and edit distance identified almost the same number of duplicates, while Jaro–Winkler metric found about 2.5 times less duplicates. This might be due to the fact that Jaro–Winkler distance is more suitable for shorter strings, such as names. Nevertheless, due to the nature of the equal fields boolean distance, we have proven that there are at least 318,818 duplicates in the processed collection. Finally, the time needed for deduplication varies depending on the type of distance metric used, equal fields boolean distance was more that 190 times faster than edit distance.

4. Conclusions and future work

It has been shown that the open access repositories collected by OAI contain many duplicates. Based on the experimental results, it can be estimated that duplicates pose at least 6.83% of the entire collection. Moreover, a generic workflow suitable for deduplication of bibliographic record and based on the open source tools has been described. It should be noted that these tools are suitable for processing relatively small collections of metadata, with ten million records being a reasonable limit. For very large collections containing hundreds of millions of records, the blocking methods available in FRIL would require a machine with a lot of RAM memory and the whole process would take a long time to finish.

In the future work we plan to harvest and deduplicate all metadata records indexed be OAI. Such task is a big data challenge and needs to be solved with a dedicated software tools such as Apache Hadoop – an open source implementation of MapReduce paradigm.

References

[1] Budapest Open Access Initiative, Budapest Open Access Initiative Home Page, March 2013, available at: http://www.opensocietyfoundations.org/openaccess.
[2] Camelot Consulting, The Link King. Record linkage and consolidation software, March 2013, available at: http://www.the-link-king.com/index.html.
[3] CDC's Division of Cancer Prevention and Control in support. Link Plus, March 2013, available at: http://www.cdc.gov/cancer/npcr/tools/registryplus/index.htm.

[4] W. Cohen, P. Ravikumar and S. Fienberg, A comparison of string metrics for matching names and records, in: *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, August 2003, pp. 73–78.
[5] DCMI Usage Board, DCMI metadata terms, March 2013, available at: http://dublincore.org/documents/dcmi-terms/.
[6] A. Elmagarmid, P. Ipeirotis and V. Verykios, Duplicate record detection: A survey, *IEEE Transactions on Knowledge and Data Engineering* **19**(1) (2007), 1–16.
[7] I.P. Fellegi and A.B. Sunter, A theory for record linkage, *Journal of the American Statistical Association* **64** (1969), 1183–1210.
[8] A. Halevy, A. Rajaraman and J. Ordille, Data integration: the teenage years, in: *VLDB'2006: Proceedings of the 32nd International Conference on Very Large Data Bases*, VLDB Endowment, 2006, pp. 9–16.
[9] J.R. Jacobs, Finding words that sound alike. The SOUNDEX algorithm, in: *Byte 7*, 1982, pp. 473–474.
[10] P. Jurczyk, Fine-grained record integration and linkage tool, Tutorial. V3.2, August 2009, available at: http://fril.sourceforge.net/FRIL-Tutorial-3.2.pdf.
[11] P. Jurczyk, Fine-grained record integration and linkage tool, March 2013, available at: http://fril.sourceforge.net/.
[12] P. Jurczyk, J.J. Lu, L. Xiong, J.D. Cragan and A. Correa, FRIL: A tool for comparative record linkage, in: *AMIA Symposium*, 2008, pp. 440–444.
[13] G. Navarro, A guided tour to approximate string matching, *ACM Computing Surveys* **33**(1) (2001), 31–88.
[14] M. Nelson, C. Lagoze, H. Van de Sompel and S. Warner, The Open Archives Initiative Protocol for Metadata Harvesting, Technical report, Open Archives Initiative, 2008.
[15] OCLC, OAIHarvester2, March 2013, available at: http://www.oclc.org/research/activities/past/orprojects/harvester2/harvester2.htm.
[16] Open Archives Initiative, Registered data providers, March 2013, available at: http://www.openarchives.org/Register/BrowseSites.
[17] Openarchives.eu, The European guide to OAI–PMH compliant digital repositories in the world, March 2013, available at: http://www.openarchives.eu/.
[18] E.H. Porter and W.E. Winkler, Approximate string comparison and its effect on an advanced record linkage system, in: *Advanced Record Linkage System*, Research report, US Bureau of the Census, 1997, pp. 190–199.
[19] A. Powell, M. Nilsson, A. Naeve and P. Johnston, Dublin core metadata initiative – Abstract model, 2005, White Paper, available at: http://dublincore.org/documents/abstract-model.
[20] Scientific Commons, ScientificCommons.org A worldwide service and registry, March 2013, available at: http://en.scientificcommons.org/.
[21] W.E. Winkler, Overview of record linkage and current research directions, Technical report, US Census Bureau, February 2006.