Matching: Discrimination, Misinformation and Sudden Death

Gail Thornburg
OCLC Online Computer Library Center, Dublin, Ohio, USA

thornbug@oclc.org

Abstract

This paper discusses the conceptual challenges faced in designing a new system of matching incoming records for a very large database from diverse sources. Problems of satisfying a "match" with sufficient flexibility and rigor in an environment of imperfect data are outlined.

Keywords: knowledge representation, software design, information retrieval, matching, database finding duplicates in large databases, misinformation

Introduction

Sameness is a sometime thing. Libraries and other information-intensive organizations have long faced the problem of large collections of records growing incrementally. With the computerized records in a networked environment has come the recognition that duplicate records pose a threat to effective information retrieval.

Yet what constitutes a match may be neither exact nor duplicate. Levels of discernment are required to permit matches on records that do not differ significantly and those which do.

Initial Definitions

What is matching? For this discussion, matching is the process by which additions to a large database are screened and compared with existing database records. Ideally this insures that duplicates are not added, nor erroneous replacements made of records pairs that are not really equivalent. A detailed review of the literature in this area is beyond the scope of this paper, but sources such as O'Neill, Rogers, and Oskins, (1993) and Hickey (1979) are good overviews of the problems of identifying duplicates and the implications for matching software.

Which database? This project took place at OCLC Online Computer Library Center Inc., a nonprofit organization serving member libraries and related institutions throughout the world. OCLC's Extended WorldCat (XWC) is the database. It is the chief database capital of the organization, and it is 'owned' in a sense by the member libraries worldwide that use and contribute to it. At this writing it contains some 58 million records.

What are the database contents? Individual records in XWC are complex bibliographic representations of physical or virtual objects – books, films, URLs, maps, slides, and much more. The records use the MARC communications format (Library of Congress, 2002). For example a record for a book might typically contain such fields for author, title, publisher, date, but many more in addition. The representation of any one object can be quite complex, containing scores of fields and subfields. Such a record may be brief, or several thousand characters long. The depth and richness of the records varies enormously.

Why is matching a challenge? Does it not sound simple for computer software to compare records one to one and detect if they are identical? Perhaps so, but this is not the task of matching. Determination of what constitutes a match, under what conditions, involves a complex process of knowledge elicitation, requirements gathering, and even experimentation. Deliberating on sameness and difference in specific cases was not a trivial task. Two records describing the same intellectual creation or work (e.g. Shakespeare's *Othello*) can vary by physical form and other attributes. Two records describing both the same work and exactly the same form can differ from each other if the records were created under different rules of record description (cataloging). Two records intended to describe the same object can vary unintentionally if typographical or other entry errors are present in one or both. Thus sorting out significant from insignificant changes was a major task in developing requirements for the design of the software.

This paper discusses the problems of matching encountered in the Metadata Capture Project at OCLC Online Computer Center. Metadata Capture is essentially the new implementation of a system for processing incoming records received from institutions, once pre-processed records by the existing infrastructure, and deciding which are already in the database. Matching decisions can result in adds or replaces of records in the database, or merely flags set to indicate library holdings. Tens of millions of records are processed through the existing record loading system every year. The task of Metadata Capture Project was to redesign the system software for interaction with the new Extended WorldCat database. Matching was one of the chief subprojects in the two year venture.

Key project aspects to be discussed here:

- Scenario and Scope -- The goals of the project
- Constraints
- Risks
- Rules and Heuristics
- What was accomplished
- Future directions

Scenario and Scope

The OCLC WorldCat database was long established, but the environment quite new. In this project there were substantial changes at the levels of database software, indexes, operating systems, and application programming languages.

The goals of the project were to write matching software suitable for processing large numbers of incoming records, representing diverse types of materials, against a new version of the OCLC Extended WorldCat (XWC), an Oracle database environment with new features and new opportunities. The old matching software used against the in-house database developed on the Tandem platform was not usable for this task.

Note: this was not a port of existing software to a new language and operating system, but redesign of matching to suit the current structure and contents of XWC and the new opportunities available there. It was also intended to address historical limits of the previous system. The existing WorldCat matching had been designed in an era when WorldCat chiefly held records about books. That composition has changed dramatically over time. A fresh look at the matching design was indicated.

At the same time the software needed to accept and convert records which had been preprocessed in the old infrastructure, and return results readable by the old database environment. The context and constraints of this environment will be elaborated later.

Matching

Matching was one subproject, generally acknowledged to be the riskiest, within the larger Metadata Capture Project [MCP]. The scope of the MCP was to take preprocessed records into the MCP environment, hand them over to units of work including matching, record resolution (choice among multiple matches), merging of records, and updates to the database. These other units of work were substantial efforts in themselves, but are outside the scope of this paper.

For each institutional project a profile was set up to indicate what was to happen to records, for instance whether only database holdings were to be set, or merges and replaces might be done. Matching needed to read the profile settings to interpret its job correctly for a given collection of records. Evaluation rules were influenced in part by the profile settings.

Constraints

Early requirements did not permit early design. They were necessarily sparse pending further analysis, and some of this further analysis depended on preliminary tests. The technical team was operating in a climate of sparse design in the development cycle. Not everyone was suited to work in this conjectural mode.

Resource scarcity and deadline were among the constraints. Resources included both staff and database access, and availability of both added to deadline pressures. The full scale test database comparable in size/contents to production XWC was delayed by many weeks. The deadlines were not extended correspondingly.

Because of the long lead time for the full scale database to be available, it was necessary to implement a smaller test database for initial work to proceed. This was about 800,000 records, in comparison with the more than 52 million records in the full scale database. However no methodology for creating such a database was available, and so we had to develop our own. A stratified incoming test set had been carefully developed to represent diverse types of records. This set was used to ‘seed’ the test database with a fraction of the derived title key matches retrieved from WorldCat. Derived title keys are quite general so they were expected to contribute background noise. Occasional incoming records were consciously omitted from the test database to allow checks of appropriate non-retrieval.

Other constraints included availability of the types of indexes needed for searching by the matching software. These were being designed and implemented in the same time frame that matching was being implemented, which made for challenges in inter-team cooperation. Little was documented due to the pace of parallel database development, so tests of features proceeded in discovery mode.

One requirement was that the new matching software perform at least “as well as” the old Batchload system. However it was quite late in the project cycle before new and old results could even be compared. Moreover the details of the old matching process were not available to inspect as were the new, so differences in results could be tricky to interpret. And performance issues could not be tested at all till very late in the development cycle.

There were a few other factors that complicated the matching challenges:

- OCLC is a multi-national cooperative and there is no universal set standards and rules for creating catalog records. Differences may arise due to language or nationality of the cataloging agency contributing records.
- Record creators have varying levels of expertise.
- Rules of cataloging most widely used (Anglo-American Cataloguing Rules [AACR2], 2002) are not absolutely prescriptive and are designed to allow local deviation to meet local needs.

Risks

Matching-Specific Risks

There were several broad categories of risk specific to matching:

- Bad matches. For record replaces or merges of multiple records into one, bad matches could compromise existing database records.
- Missed matches could lead to duplicates being added to the database.
- Correct but too-slow processing was unacceptable for the everyday work environment. Extended matching was considerably slower than matching on unique record ids (details below).
- Predicting growth – the matching team needed to plan for new types of records not necessarily similar to current input, but had insufficient data to use for testing or review.
- Reviewer feedback – the review of test results was quite labor intensive and required high levels of expertise. With the numerous test cycles planned for matching, there was risk that problems might be missed, or detected late in the schedule. In addition reviewer comments led to requirements changes in many cases, which challenged the developers.
- Matching as User -- matching software was in many cases the first ‘user’ to exercise facets of database functionality. The first user of new systems tends to encounter more problems.

Project Risks

- Schedule risks – database and index availability was a major risk. If a test environment was unavailable for a week, progress could be completely stalled. Likewise if indexing schedules slipped, matching software using those indexes was affected.
- Resources – some of the staff involved were knowledgeable of the bibliographic environment, but less so of Unix, Java, and Oracle. Others were knowledgeable about Java and Oracle, but had little understanding of the database records or of the goals of the organization
- The Dynamic Systems Development Method (DSDM), described as an agile project/product development methodology, was an interesting approach (DSDM Consor-

tium, n.d.). However it was not certain that DSDM was an entirely good fit for a project of the duration, complexity, and risk of Metadata Capture.

Enterprise Risks

- The Oracle database environment was relatively new to the Division. Sometimes problems in the implementation were integral enough that a fix from Oracle itself was required.

Cooperation and Communication Factors

What stood out in the matching project was the high level of inter-team dependency. For the indexes to be implemented for the use of Metadata Capture, the teams had to cooperate to define what was needed, such as which indexes. What was needed for other users of XWC did not necessarily line up with the needs of a system such as MCP.

Moreover close coordination was required with the Oracle database administrators to insure that the search software used by matching was reasonable and efficient for the Oracle features being implemented. At times this went as far as tailoring and fixes from Oracle itself; OCLC is currently one of the biggest Oracle Text users that Oracle Corporation has.

A whole category of coordination became necessary for users of the test environment. One team’s baseline performance test might be compromised by the matching team running extensive tests at the same time, or the matching team might lose carefully planned test results if another team bounced the environment without notice. A performance listserver was set up to announce planned events, which reduced sudden team agitation considerably.

For a bibliographic utility, influence from member institutions is always a factor of importance. If a member institution or group felt that certain features of the new implementation were contrary to library/user interests, OCLC needed to be responsive.

Though version control systems were in place for the software, the requirements, and the design documents, their numbers tended to inundate the team with revision work. Yet this was critical to the sustainability of the project over time.

Throughout the project the tone of the matching effort was one extreme focus, at times suspension of disbelief. Whatever else was ‘in flames’ to the right or left of matching had to be ignored.

Overview of the Matching Process

Matching was called from the larger MCP environment, with settings specific to a given institutional project. A gatekeeper or driver program called the matching components as needed for a given profile.

Unique Key Matching

The previous system of matching had four unique keys: OCLC number, Library of Congress Control Number [LCCN], ISBN, and ISSN.

The move to XWC allowed the use of eleven keys for unique key matching. These included the keys above, as well as URI, Other System Control Number, Publisher Number, and others.

The incoming record was evaluated. All unique keys profiled to be searched, for which a search key could be formed, were packaged together in an OR Boolean construction, and sent together to the database. A maximum of 7 results was set for any unique key search. Most key searches did not return more than a few hits, but occasionally a larger number could be returned. If the number exceeded 7, the set of matches was discarded without retrieving the records; those were not 'unique' enough. The record(s) retrieved constituted a candidate set to be verified after unique key matching.

Evaluation

Following unique key matching, qualifier checking might be done with the candidate(s). This involved comparisons of date, derived title check, and material type. These were safety checks; a contributor might have 'clone' a database record to create a new one, but not cleared out all the old data. With these checks, candidate matches might be retained or rejected. Preliminary evaluation assessed the need to stop or continue matching. This decision depended on several factors both project and record specific. For example if database records were to be replaced, matching needed to be more careful and more extensive than if the goal of the project was merely to set holdings (indicating that a library held an item).

Extended Matching

Extended matching was intended to cover the complex or uncertain evaluations, as well as further searching of XWC. The candidate set of records obtained was ranked, then passed to extended matching. If a decision had not been made to stop, the first step was comparison point evaluation (see below) of the unique matching candidate set. If this step ruled out all candidates, new searches of the database would be formulated and executed. Then comparison point evaluation would be executed on each of these.

Ranking the Candidate Records

One difference between the old Batchload and the new MCP was the retention of unique key matches. In the older system, if the matching could not be resolved to 1 record at the unique key stage, the results were discarded before extended matching. In the new system, these unconfirmed matches were retained and passed into extended matching; their more detailed evaluation was the first phase of extended matching.

Software was developed to rank the existing candidates by the number of keys on which matches were obtained. For instance, if all 11 keys were profiled for a given file, and one candidate matched on three unique keys, while a second matched on only one, the goal was to give the multiple-match database record a higher rank than the singly-matched one. Conceptually each match on a key was one row in a table of the matches.

Comparison Point Evaluation

Given candidate matches [from unique or extended matching searches], comparison was made between incoming record and database candidate, on a set of features. These proceeded roughly in order of more significant to less. A clear mismatch on any one comparison point ended the evaluation of that candidate. This was fairly ruthless. (A new search that re-found the same record would not consider that record again.)

Query Formulation and Search

If comparison point evaluation of the Unique Key matches ruled out all candidates, it was necessary to conduct further searches of the database. At this point new queries needed to be formulated. Early in the design it was decided that a maximum of three new searches would be attempted against XWC. These could be chosen from a set of six possible queries, depending on factors of the individual match situation.

Key Challenges for Extended Matching

Detailed comparisons and judgments on inexact matches are the province of extended matching. The key is to set the barriers to matching loosely enough to permit analyses of near-miss cases, but define the requirements for match status carefully enough to disallow clear non-matches.

Searching the new XWC

How to formulate good queries for the new extended matching was an early concern. We had no definite method for deriving them. Search terms were derived from fields in the incoming records and combined to form Boolean queries used to search the database. We knew the formulation of queries for searches against XWC had to be both general-purpose and robust. Yet the search process had to be sensitive to peculiarities that could bog searches down.

The first formulation examined the incoming record and selected a search term (excluding stop-words) from the title, author, publisher, date, a second title term, and a type of material or format. The query terms were and-ed together in various combinations to form a Boolean search.

If a search found too many results (more than 20 for extended matching), it was added to the dynamic list of searches already "known too large" and avoided in future searches within the same file. This was useful due to the nature of the sets of records received, which might contain groups of very similar records.

This query formulation worked well in many cases. Yet results might vary from zero results in search one, to very high number of results in the very next search – or vice versa. It was quite difficult to select terms and predict their specificity. This approach was used until the transition from testing against our prototype database (800,000+ records) to the full scale copy of XWC. At this point the numbers of too-large searches overwhelmed the team's expectations.

Numerous trial approaches showed little progress in addressing the too-large-results problem. There seemed to be no observable patterns. Then we tried deriving a title search using proximity searching operators, and at the same time sent all the query terms available in the first search. This reduced the number of unusable too-large result sets to quite a tolerable number. And in the course of all these trials, we learned a lot about indexes in the new XWC. We learned it was very hard to generalize about good searches.

Domain Rules – Knowledge Elicitation

Early Domain Modeling

Two main types of domain expertise were needed to drive the matching software through the process with some semblance of intelligence. One was when to consider matching complete, the second was to know when to go on, that is to continue beyond unique key matching and into extended matching.

Decisions to continue to extended matching were the focus of initial meetings on rules development. While the profile setting for an institution's file might indicate extended matching, the process should still usually stop if a single unique key match was confirmed. Since extended matching would necessarily be slower than Unique Key matching, it was important to identify just when it was really needed.

The domain experts met with the development team and drafted rules on paper to describe situations where extended matching should be invoked. The group examined all conditions painstakingly to try to determine the necessity and sufficiency of the combinations. This proved hard work, and tended to proceed in cycles. List of rules were re-examined at intervals, as more and more of the software was implemented. These were simple if-then constructs dependent on order, first rule firing wins. An example of the form used in the rules sessions follows:

```
If      OCLC number matches, AND
        There is a single match, AND
        Qualifier checks are confirmed, AND
        Not profiled to replace,
Then    Matching is complete.
```

Related Rules Considerations

Rules for declaring candidate mismatches were encapsulated within the comparison points evaluation software. For example, if data for one comparison was available in the incoming but not the database match, the rule was often to declare an 'unconfirmed' status and go on to the next point.

Extended matching search formulation had developed in what was a relatively format-neutral approach. Type of material was given less weight in the searches. What evolved could be described, format-wise, as searching loosely, discriminating closely. This meant the rules deciding to stop before or after comparison points were evaluated had an interesting impact on matches achieved. For some situations, stopping before comparison point evaluation had the effect of suppressing detail-oriented mismatches – and this was quite acceptable.

Streamlining the Rules

Some rule conditions were fairly simple to check, but others involved considerable comparison and computation. We realized in cycles of testing that the simplest rules should ideally be considered as early as possible in the matching process, hopefully before unnecessary calculations were undertaken. At the same time, rules which seemed to carry more uncertainty gradually shifted lower in the ordering. These refinements were easier to identify in retrospect than to predict.

Discrimination

With so much to be clarified in the initial specification, the requirements for comparison points between incoming and candidate records were late in development. This was one type of discrimination exercised in matching. Comparison point evaluations returned a status of "match" if satisfied, "mismatch" if not. We dealt with the uncertainty of missing or ambiguous information in one of the record pairs by assigning a status of "unconfirmed."

One of the reasons the numerous requirements for comparison points worked so well, with so little time, was the focus on discrimination. If each requirements document gave examples of mismatch as well as match, they tended to be stronger descriptions for the development team.

Material Type

A particularly thorny issue arose in development of the requirements for material type evaluation. This dealt with format considerations, such as avoiding mismatches of a book with a video, or a

VHS video with a Beta format or DVD. (See, for example, Weitz, 2001, for a discussion of the challenges of cataloging videorecording, and a bibliography of work published in this area.) The indexing teams for XWC had developed detailed requirements for deriving material types from the information in a record being indexed. The decision to rely on these types for matching was a necessary synchronization with indexing. The process of evaluating a given record and inferring the material type of the object represented was excruciatingly complex.

Nonetheless developing requirements for determining match between the identically derived material types in the incoming record and the database candidate was a big job. Due to overlaps in meaning, and potential ambiguities in the records, the material type values in incoming and database record could not simply be compared. These were multilevel and not equally significant. Some valid values meant nothing to matching. So devising an effective scheme of ordered comparisons was a conceptual challenge.

Sudden Death

The series of comparison points for a given record pair could terminate at any mismatch. As we developed the framework for running comparisons, we theorized that the most discriminatory of comparison points could be predicted, and ordered. So those points considered most telling were at the top of the list. Candidate elimination sometimes went rapidly if ruthlessly.

The qualifier checks (date, title key, material type) applied to unique key matching could result in quick elimination of candidates too. This was significant for extended matching because the mismatch list was retained, and a record re-discovered in extended searching would not be reconsidered if on the mismatch list.

Cause of Death

The test cycles could track just where a record match was rejected. (See Appendix One for examples of mismatch status.) With three qualifier comparisons for both unique and extended matching, and over a dozen for Extended, results would have been very difficult to trouble shoot without clear tracking of which comparison point module was the first to shout "mismatch"! The comparison point requirements were in flux at this point, and needed a log of events for matching tests to be evaluated.

Of course the team's early communication on comparison point requirements showed gaps and omissions. The event was recorded in our log (See Appendix). This sort of event was effective in getting the reviewer's attention.

Violated Expectations: Handling Misinformation in Matching

Expectations about the nature of records in the databases were frequently violated. What seemed to be good rules for matching might not work well if the incoming data was not well formed. Background discussion of broader misinformation issues in shared library catalogs can be found in (Bade, 2002), and a good though dated review of duplicate record problems can be found in (O'Neill, et al., 1993).

Sources of bad or confounding information found by the Matching team included the following:

1. Historical independence. Prior to the 1970's, most libraries did not share their cataloging with other libraries. Many institutions, especially smaller ones, were outside the loop and did things their own way. They used what rules they felt were useful and if there were rules at all. Later they converted sparse and poorly formed data into MARC records and sent them to OCLC for matching, perhaps in an effort to get back a more complete and

useful record. Yet the matching process could not always distinguish or interpret these local dialects.

2. Varied standards worldwide. While U.S. libraries usually follow AACR2 and use the MARC21 communications format, other parts of the world may use UNIMARC and country-specific cataloging rules. Some may not use any form of MARC but submit a spreadsheet that is then converted to MARC. There is some potential for ambiguities in those conversions due to lack of 1:1 correspondence of parts.
3. Typographical errors in titles and other parts of the record. Anywhere the software had to parse text, an entry error – or even correction of an entry error by a later update – could confound matching. This could confound both a) query execution and b) candidate comparisons.

4. Typographical errors in titles and other parts of the record. Anywhere the software had to parse text, an entry error – or even correction of an entry error by a later update – could confound matching. This could confound both a) query execution and b) candidate comparisons.

5. Errors of formatting of variable fields. The rules for data entry in the MARC record are complex, and have changed over time. Erroneous placement or coding of subfields posed challenges for identification of relevant data. The requirements had to be refined to be fault tolerant wherever possible.
6. Errors in format information. This can affect Material Type derived for the record.
7. Bias toward less generic titles in matching. Because the limits of processing mandated a limit on the size of result set Matching would analyze, retrieval in Extended Matching could tend to discriminate against generic titles.
8. Language of cataloging. This comparison had in the past caused inappropriate mismatches. The requirements in the new matching aimed to address this.
9. Language in formation of queries. This was not considered in the design, in part because of many complexities in identifying the language of the record and item being described. MARC records frequently are a mixture of languages. As has been seen in other projects with intensive comparison of text, overlap in languages has the potential to confuse comparisons of short strings of text (Thornburg, 2002). The assumption in this project was that the use of all possible syllables in the title would tend to mitigate language problems.
10. Fixes to cataloging errors have the potential of preventing a match to what was actually the same record. If a cataloger cleaning up a record fixes a typo, an attempted match later may fail if the incoming record is unchanged.

Checklist

The stresses to software development projects are well known: deadlines set in advance of schedules, external pressures, late requirements, scarce resources. Most teams have to deal with these. Reflection on a project cycle as complex as matching does suggest there are general types of challenges, and sometimes practical ways to deal with such stresses. To name a few…

Requirements Uncertainty

Focus on what the domain experts can agree on. Well designed test cycles should identify problem areas for resolution. The rest of the disagreements should be looked on as potential for later enhancements. That is, experts often defer to each other's respective areas of knowledge. Differences of opinion may point to interesting overlaps which could enrich or simplify the software design.

Learning from Examples

This is a telling technique – good, but expensive learning. While examples are essential, counter examples make it easier.

Discrimination

What prunes the search tree? For this team it was helpful to focus on discriminating examples of good rules, not just descriptive rules. Counter examples were effective in fleshing out rules and requirements.

Satisficing

Focus first on a rule that will attack the 80% first. Document its limits, and specialize it once that first version is working. Recognize that the complexity and uncertainties of the domain make optimal solutions unlikely to achieve.

Choosing Battles

We knew that some problems would be intractable in our first cycles of tests. The problems of generic titles for instance needed to be set aside until we could evaluate overall performance. Some of these proved to be cases where unique key matching could ameliorate the problem. This was not to suggest ignoring it, but meant we could defer further analysis until later. Not everyone can consciously distance themselves from the details, and some very knowledgeable professionals could have failed to cope with this project because of a too-early commitment to explicating details.

Eliciting Rules

Focus on the hard work of identifying all necessary clauses to a rule, and ignore order at first. The ordering / weighting of rules will be easier to refine in testing once the key rules are in place.

Testing Opportunities

For a mission-critical project, it is important to have an aggressive champion, someone unafraid to fight for test cycles. There tends to be competition for testing space. The matching algorithms and software were being developed at the same time other aspects of database development such as indexing and optimizing.

Incompleteness

Of test results: in a large changing database environment, one cannot exhaustively define all of the right and wrong answer(s) for a test set. Nor will comprehensive evaluation of test results be possible in some cases. Figure out a plan for doing without, but tests with all the unusual data you can find.

The Big Picture

There is an argument for the necessity of 'tunnel vision' when working under pressure, but there is a cost. In this project a lack of realization among teams of their interdependencies set us back many times. Communications improved as the project went on, but the importance of cross-team and cross-management communication cannot be over stressed. We would have addressed it more aggressively earlier in the process had we known, yet it is a credit to the teams involved that all pulled through.

Attitudes and Personality

- Sense of humor: this was as important as a sense of teamwork, which itself was critical in a project of such uncertain scope and such certain stresses.
- Willingness to Experiment: If nothing blew up, the tests probably weren't varied enough, or rigorous enough. Or the problem wasn't interesting enough.
- Ability to cope with non linear progress: Sometimes the team had take a step back, or sideways, or even duck and weave.

Conclusion – What Was Accomplished

The matching team started out with a very general set of requirements for Matching, little knowledge of how the new Oracle XWC database would really work, and whether it was possible in the time frame allowed to reinvent Matching for a new generation database.

By the install of Version One matching, the team had succeeded in implementing workable, well-tested software which accomplished the following, and more:

- The matching infrastructure achieved a format-neutral design that permitted use of one set of programs for all types of materials. Material type comparisons were made at the point they were necessary, and a level of abstraction away from the details of individual record encodings was achieved.
- Unique key matching was made possible on far more keys than the old system had at its disposal.
- Extended matching could be invoked intelligently and with reasonable efficiency. It spared the results of unique key matching, unlike the old system. Generally it was as successful in matching as the old system, and in some areas there were measurable improvements.
- The new system provided a base from which to expand and improve. The system and requirements were better documented than the preceding system. Batchload, while basically reliable, was not fully designed and documented. It was developed while contracts were being negotiated to use it—; so there was not enough time for careful design and no time for documentation.
- A test environment was not available except at a very small level in Batchload. This was designed into the new system.
- A design approach that focused on discrimination and counter-examples proved a useful tool for team communication. Requirements were clearer and software design was easier.
- Sudden death as design decision: rejecting candidates immediately on one point of mismatch was surprisingly effective overall. This suggested that the rules for distinguishing

match from mismatch had been successfully encapsulated at a lower level than the comparison point framework. Anything the domain experts didn't like resulted in a change to the software for a comparison point, not a change to the framework.

Future Directions

There are some intriguing areas that could not be addressed in the scope of the first version matching in Metadata Capture.

- Material types should be explored. There may be areas where we can achieve even more desired matching across formats. Input errors in material type: can we guard against those with use of internal evidence?
- Differing opinions among domain experts. – can we mine these for matching enhancements?
- Title comparison issues we will always have with us. Can we increase the tolerance of the software for minor faults, without bogging down performance generally?
- Are there ways the literal-minded comparison point mismatches can be made to suspend judgment where advantageous to increased matching ? Are there ways to exploit the comparison point framework, now that the comparison point objects have been pushed down? It is a clear and fertile field. Would experiments with weighting the points in the framework be worth the added complexity
- New types of non-MARC records – we know they are coming. How will they affect the heuristics of matching?

References

Anglo-American Cataloguing Rules [AACR2] (2nd ed., 2002 Revision). (2002). Chicago: American Library Association.

Bade, D. (2002). The creation and persistence of misinformation in shared library catalogs. Occasional Paper No. 211, April 2002. Graduate School of Library and Information Science, University of Illinois at Urbana –Champaign.

DSDM Consortium. (n.d.) *Delivering agile business solutions on time*. Retrieved November 21, 2004 from http://www.dsdm.org/tour/default.asp

Hickey, T. B., & Rypka. D. J. (1979). Automatic detection of duplicate monographic records. *Journal of Library Automation, 12* (2) June, 125-142.

Library of Congress. (2002). *MARC 21 concise format for bibliographic data*. Retrieved November 20, 2004 from http://www.loc.gov/marc/bibliographic/ecbdhome.html

O'Neill, E. T., Rogers, S. A., & Oskins, W. M. (1993). Characteristics of duplicate records in OCLC's online union catalog. *Library Resources and Technical Services, 37* (1), 59-71.

Thornburg, G. (2002). The syllables in the haystack: Technical challenges of non-Chinese in a Wade-Giles to Pinyin conversion. *Information Technology and Libraries, 21* (3), 120-126.

Weitz, J. (2001). Videorecording cataloging: Problems and pointers. *Cataloging and Classification Quarterly, 31*(2), 53-83.

Appendix

The following is an example of a test report for one incoming record as processed through matching. Each matched on 3 unique key searches: nbacn, isbn, and oscn. The process found two candidate matches via unique key matching and evaluated both, rejecting 36423286 on the **Publisher [cPub]** comparison point. The record retained is summarized in the <Match> section at the bottom of the report.

```
<Results>
   <BriefBRec>
   <BRecNo>13</BRecNo>
      <c00080711>1996</c0080711>
      <v020a>2550301129</v020a>
      <v029a>NLC</v029a><v029b>96802503X</v029b>
      <v035a>[]</v035a>
      <v245_in>Québec biodiversity strategy, summary.</v245_in>
      <v260b>Gouvernement du Québec, Ministère de l'environnement et de la faune,</v260b>
   </BriefBRec>

     <ResultArray>
        <ResStr>61314776,1 of 2,,Québec bi,(ba:"96802503X" and bg:a and
bj:NLC),nbacn,match found</ResStr>
        <ResStr>61314776,2 of 2,,Québec bi,bn:255030112*,isbn,match found</ResStr>
        <ResStr>61314776,1 of 2,,Québec bi,qa="NLC 96802503X",oscn,match found</ResStr>
        <ResStr>61314776,1 of 1,00000000,          , ,date, - match</ResStr>
        <ResStr>61314776,1 of 1,00000000,          , ,title, - match</ResStr>
        <ResStr>61314776,1 of 1,[],   ,     ,mattype, - match pri/pri </ResStr>
        <ResStr>61314776, 1 of 1 ,[],         , ,ctitle,- match</ResStr>
        <ResStr>61314776, 1 of 1 ,[],          , ,cPub,- match</ResStr>
        <ResStr>61314776, 1 of 1 ,[],         , ,clangcat,- match</ResStr>
        <ResStr>61314776, 1 of 1 ,[],         , ,csize,- match</ResStr>
        <ResStr>61314776, 1 of 1,[],           ,  ,cLCCN no LCCN,- unconfirmed</ResStr>
        <ResStr>61314776, 1 of 1 ,[],          , ,cPubPlace,- match</ResStr>
        <ResStr>61314776, 1 of 1 ,[],          , ,cExtent,- match</ResStr>
        <ResStr>00000000,0 of 0,,          ,,lccn,no search term found</ResStr>
        <ResStr>00000000,0 of 0,,          , ,repno,no search term found</ResStr>
        <ResStr>00000000,0 of 0,,          ,,oclc,no search term found</ResStr>
        <ResStr>00000000,0 of 0,,          , ,issn,no search term found</ResStr>
        <ResStr>00000000,0 of 0,,          , ,uri,no search term found</ResStr>
        <ResStr>00000000,0 of 0,,          , ,pubno,no search term found</ResStr>
        <ResStr>00000000,0 of 0,,          , ,osn,no search term found</ResStr>
        <ResStr>00000000,0 of 0,,          , ,coden,no search term found</ResStr>
        <ResStr>00000000, 0 of 0, 00000000, ----------,eval,Eval rule fired is number:
9.0</ResStr>
        <ResStr>00000000,0 of 0,00000000,           ,Ext,Comparison pts confirm
match(es)</ResStr>
        <ResStr>36423286,2 of 2,,Québec bi,(ba:"96802503X" and bg:a and
bj:NLC),nbacn,match found</ResStr>
        <ResStr>36423286,1 of 2,,Québec bi,bn:255030112*,isbn,match found</ResStr>
        <ResStr>36423286,2 of 2,,Québec bi,qa="NLC 96802503X",oscn,match found</ResStr>
        <ResStr>36423286,1 of 1,00000000,          , ,date, - match</ResStr>
        <ResStr>36423286,1 of 1,00000000,          , ,title, - unconfirmed</ResStr>
        <ResStr>36423286,1 of 1,[],   ,     ,mattype, - match pri/pri </ResStr>
        <ResStr>36423286, 1 of 1 ,[],         , ,ctitle,- match</ResStr>
        <ResStr>36423286, 1 of 1 ,[],          , ,cPub,- mismatch</ResStr>
     </ResultArray>

     <MRecArray>
     <Match>
        <c001>61314776</c001>
      <c00080711>1996</c0080711>
      <v020a>2550301129</v020a>
        <v029a>NLC</v029a><v029b>96802503X</v029b>
      <v245>Québec biodiversity strategy, summary.</v245>
        <v260b>Gouvernement du Québec, Ministère de l'environnement et de la
faune,</v260b>
     </Match>
     </MRecArray>
</Results>
```

Biography

Since completing her doctorate in information science from the University of Illinois at Urbana-Champaign, **Gail Thornburg** has held diverse positions in academia and the larger information community. She has taught at the University of Maryland and the University of Illinois, and served as an adjunct professor at Kent State University. She has worked as a prototype developer, a network and database administrator, and most recently as a senior level software engineer at the nonprofit bibliographic utility OCLC, Inc.