Using Statistical Models for Dynamic Validation of a Metadata Extraction System

Kurt J. Maly, Steven J. Zeil, Mohammad Zubair
Ashraf Amrou, Ali Aazhar, and Naveen Ratkal
Old Dominion University
Dept. of Computer Science
Norfolk, VA 23529
dtic@cs.odu.edu

January, 2007

Abstract

A dynamic validation process is described for an application (metadata extraction from scanned documents) where a moderate failure rate is acceptable provided that instances of failures during operation could be identified. Lacking a plausible exact oracle for the application, a series of statistical models of output characteristics is employed. Flexibility and adaptability is achieved by developing a customized scripting language describing how the various tests should be combined to obtain an overall measure of confidence in a program output. The suitability of the validator was demonstrated by an experiment measuring its ability to mimic human judgments as to which of several alternate outputs for the same document would be preferred.

1 Introduction

The authors have been involved in the design and development of a system to extract metadata information from scanned documents stored in government repositories including collections at DTIC [6], NASA [10], and the GPO [16]. These collections are large (e.g., the DTIC collection contains more than one million documents and adds tens of thousands of new documents each year. The documents are diverse, including scientific articles, slides from presentations, PhD theses, (entire) conference proceedings, promotional brochures, public laws, and acts of Congress. Contributions to each collection come from a wide variety of organizations, each with their own in-house standards for layout and format, so, even among documents of similar kind, the layouts vary widely. The amount of metadata available may vary considerably as well. Many documents have only "conventional" metadata such as title, author names, publishing organization, and date. Others have more specialized metadata including security/release restrictions and waivers, public law numbers, or keywords and index terms. Different collections may also target different metadata fields for storage in their databases. Figure 1, for example, shows a possible metadata record for this very article in the style of the DTIC collection. (If the collected document were to include a copy of the cover page of the proceedings, a number of additional fields might be extracted as well.)

The heterogeneity of these collections poses a challenge to the development of an automated system for metadata extraction. Our approach has been based upon a two-part process [13, 14], in which

- A new document is *classified*, assigning it to a group of documents of similar layout. A vari-

```
<metadata>
  <UnclassifiedTitle>
    Using Statistical Models for Dynamic
    Validation of a Metadata Extraction
    System
  </UnclassifiedTitle>
  <PersonalAuthor>
    Kurt J. Maly
  </PersonalAuthor>
  <PersonalAuthor>
    Steven J. Zeil
  </PersonalAuthor>
  <PersonalAuthor>
    Mohammad Zubair
  </PersonalAuthor>
  <PersonalAuthor>
    Ali Aazhar
  </PersonalAuthor>
  <PersonalAuthor>
    Ashraf Amrou
  </PersonalAuthor>
  <PersonalAuthor>
    Naveen Ratkal
  </PersonalAuthor>
  <CorporateAuthor>
    Old Dominion University
    Dept. of Computer Science
    Norfolk, VA 23529
  </CorporateAuthor>
  <Abstract>
    A dynamic validation process is...
  </Abstract>
  <DistributionStatement>
    Permission to make digital or...
  </DistributionStatement>
</metadata>
```

Figure 1: DTIC-style Metadata Record for this article

ety of techniques have been and continue to be explored for this [13], but the basic goal is to group together documents whose title or other metadata-containing pages would appear similar when viewed (by humans) from several feet away.

- Associated with each class of document layouts is a *template*, a scripted description of how to associate blocks of text in the layout with metadata fields. For example, a template might state that the text set in the largest type font in the top-half of the first page is, in that layout, the document title.

  An extraction engine interprets the template for the chosen document class, following its scripted instructions to actually locate and extract text strings that make up the values for various metadata fields.

The automated extractor is intended to replace a labor-intensive human process. As such, it could be economically viable even with failure rates on the order of 20-30% of documents having incorrectly extracted fields, *provided that* these incorrect outputs can be identified during program operation and referred for human corrective action. There are a number of potential sources of error in the system, not all of which can be directly controlled:

- The input to the system consists of scanned documents that have been passed though a commercial Optical Character Recognition (OCR) program to produce an XML version of the document text marked with formatting information.

  Errors in this OCR output are quite common. Some of these may be attributed to failures of the OCR software, but many are probably inherent in the approximate nature of the OCR process. In addition, the OCR output may be corrupted because of pages that are smudged or that have handwritten or stamped notations or images superimposed on the document text.

- The classification process is subject to failure, resulting in the selection of an inappropriate template.

- Though uncommon, it is possible for documents that appear visually to have similar layouts to actually present their metadata in different locations or in different orders. A template for that document class will therefore sometimes associate metadata values with the wrong field.

- A template may contain errors. Because templates are typically written based upon the first few encountered examples of a given layout, there is a danger of designing the template to key in on coincidental similarities that will not hold across the entire class.

- Finally, there is the possibility of internal faults in the engine that applies the templates to the OCR'd documents to actually perform the extraction.

It is in this context that the dynamic validation process described in this article was developed. The authors believe that this process may prove valuable in other applications where exact oracles are impossible or infeasible.

The next section presents a brief overview of related work in dynamic validation and the development of testing oracles. Section 3 describes the design of the validation process and Section 4 presents an experiment intended to determine if the validator is capable of making reasonable decisions as to what kinds of output are preferred for selected documents.

2 Related Research

The problem of selecting an appropriate oracle for determining the correctness of a program output is well known in testing literature and practice. Testing oracles have traditionally run the gamut from human inspection of printed outputs to automated checks for conformance with a formal statement of expected values. Common intermediate forms include comparison against outputs from related, usually older, versions of the system (back-to-back testing) and automatic comparison against human-computed "gold" outputs.

Oracles for programs that, like our metadata extractor, attempt to mimic a human activity are notoriously problematic. [17] In meta-data extraction, for example, even expert humans may differ on what is the best description of a given document. Another is that a certain tolerance may be accorded to alternative renderings (e.g., an author might be considered correct whether rendered as "John Doe" or "Mr. John Doe"). Typically, such applications are validated by allowing human experts to rate outputs as reasonable or unreasonable, e.g., [9]. Such an oracle makes sense only prior to software deployment, however. It would be self-defeating to require continuous human inspection of outputs once the software has been deployed.

Although verification and validation are typically considered as an activity prior to deployment of a software version, there is substantial precedent for continuing them throughout operation of the deployed software. This may be done as protection against unanticipated changes in the operational environment [8], as a defensive measure against undetected faults [1], or as part of a systematic structure, such as recovery blocks, intended to promote robustness [3].

The terms *dynamic verification* and *dynamic validation* have been coined to describe the provision of run-time checks as part of the running software to monitor the correctness of a calculation [1, 7]. The most common approach to dynamic verification and/or validation is based upon programmed assertions [2, 12, 18], which can provide valuable checks on the internal structure and consistency of a computation. Dynamic validation is often, though not necessarily associated with post-deployment validation – in some cases assertion checking is disabled in the final release.

Assertion checking is not a viable option for the metadata extractor, in part because the precise specification of desired behavior is not possible for the final output. Although assertion checking may add to confidence in the internal consistency of the extractor computations, the bulk of the sources of potential error are associated with inputs to the extractor and the suitability of the chosen computation for a given document.

More generally, dynamic verification may encompass a range of programmed tests that exploit internal access to the computation state to detect an impending failure, as in the acceptance tests of software recovery blocks [5, 11]. Once again, however, it is doubtful that the precision for a proper oracle could be achieved. Also questionable is the idea of whether internal computation state can be useful in contexts where the selection of the appropriate computation (template) is one of the primary sources of uncertainty.

We have therefore devised the novel approach of constructing a statistical oracle, a mechanism for examining outputs and determining to what degree we believe they are "typical" of correct metadata. The information exploited in these tests is independent of the document features used to specify the templates. Such independence is commonly regarded as desirable, if not essential, to the quality of validation [4]. The next section describes the design of our validator.

3 The Validator

In this section we present our validation techniques that we successfully applied to legacy collections from DTIC and NASA. The main idea behind our statistical validation techniques is measuring a relevant property, such as the length of the title (a metadata field), and comparing it to values known to be typical of documents already in that collection. The comparison results are quantified and normalized into a confidence value for the value extracted for that metadata field. Confidence values range from a minimum of 0.0 indicating an almost certainly incorrect value to 1.0 indicating an almost certainly correct value.

3.1 Reference Models

Most of the tests supported by our validator (detailed below in Section 3.2) are based upon a point statistic computed for a metadata field value extracted by the program and compared to models of that field. The model information is obtained from metadata records previously produced by the human staff for documents already in the collections.

Field	Avg.	Std. Dev.
UnclassifiedTitle	9.91	4.78
Abstract	114.32	58.02
PersonalAuthor	2.75	0.52
CorporateAuthor	6.99	2.29

Table 1: Field Length (in words), DTIC collection

Field	Avg.	Std. Dev.
UnclassifiedTitle	88%	13%
Abstract	94%	5.0%

Table 2: Dictionary Detection (% of recognized words), DTIC collection

For the DTIC collection, for example, the average and standard deviation of the field lengths are computed for titles, author fields, organization names, and abstracts for approximately 800,000 (unclassified) documents that were made available to us. Table 1 shows the length properties for the DTIC fields we are currently validating.

For the titles and abstracts, the average and standard deviation of the percentage of words in an English dictionary are computed for the same 800,000 documents. Table 2 shows the recognition rates for these DTIC fields.

For author and organization names, which should feature a more specialized but recurring vocabulary, phrase dictionaries are constructed for all phrases of length 1-4 words over a randomly selected set of 600,000 of those metadata records. Then the remaining 200,000 are used to compute the average and standard deviation of the percentage of phrases in each field that are recorded in the phrase dictionaries (an approximation of the percentage of phrases that, in new documents, can be expected to be recurrences of previously encountered phrases). Table 3 shows the recurrence percentages for these DTIC fields. It shows that, particularly in the corporate author field, a document will rarely introduce a word or even a phrase that has not been encountered before, suggesting that almost any output with novel content in that field would be suspect.

Similar models are computed for the NASA col-

Field	Phrase Length	Avg.	Std. Dev.
PersonalAuthor	1	97%	11%
	2	83%	32%
	3	71%	45%
CorporateAuthor	1	100%	2.0%
	2	99%	6.0%
	3	99%	10%
	4	98%	13%

Table 3: Phrase Dictionary Hit Percentage, DTIC collection

lection, although the available set of metadata is smaller, comprised of about 10,000 records.

The DTIC and NASA collections include many more metadata fields than have been mentioned here. As the extraction system moves toward final deployment, similar models will be constructed for the remaining fields.

3.2 Basic Validation Tests

Our validation tests can be categorized into two main categories: *deterministic tests* and *statistical tests*. The former include pattern matching using regular expressions and date format. The latter include the length, vocabulary, and dictionary tests described below. The pattern matching and regular expressions are straightforward but are applicable only to a relatively small fraction of metadata fields that have restricted formats. The probabilistic tests are described next.

3.2.1 Length Test

The length test compares the length of the metadata field value to an average length previously calculated for this metadata field from known correct metadata. If the value is significantly longer or shorter than typical length, it receives low confidence value. The closer the length to average, the higher the confidence score it receives.

3.2.2 Vocabulary Test

The vocabulary test is suitable for metadata fields that tend to have a specialized vocabulary, such as author and organization names. Phrase dictionaries are constructed for these metadata fields using preexisting correct metadata. A phrase dictionary contains all the sub-sequences of tokens for each metadata entry. For example, if the string is ?Kurt John Maly? then the phrase dictionary would contain the following sequences: Kurt, John, Maly, Kurt John, John Maly, Kurt John Maly. Then, when validating an extracted metadata value, it is tokenized into phrases and the corresponding phrase dictionary is consulted to count how many of these phrases exist in the phrase dictionary. The percentage of recurring phrases for this extracted value is compared to the average recurrence rate of phrases for that metadata field as determined from the preexisting correct metadata.

3.2.3 Dictionary Test

The dictionary test is suitable for metadata fields that do not have a specialized vocabulary, but rather are composed of free text, e.g., titles or abstracts.

When validating an extracted metadata value using the dictionary test, the extracted value is tokenized into words and an English language dictionary is consulted to count how many of these words exist in the dictionary. The percentage of recognized words is compared to the average percentage of recognized words for correct values of that metadata field in the preexisting metadata.

3.3 Normalization

Each of the statistical tests entails, as noted earlier, the computation of a point statistic (x) for an ex-

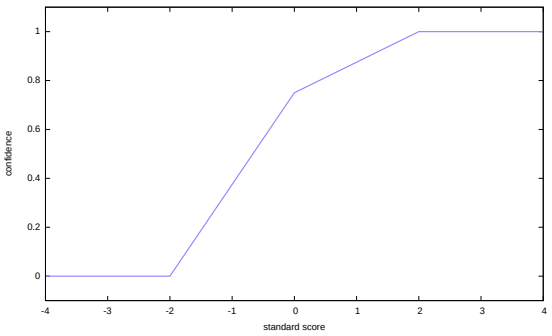
tracted metadata field. In each case, the associated model for that field supplies an average ($\bar{x}$) and standard deviation (s) for that statistic. This permits the computation of a standard score for that point statistic:

$$z = \frac{x - \bar{x}}{s}$$

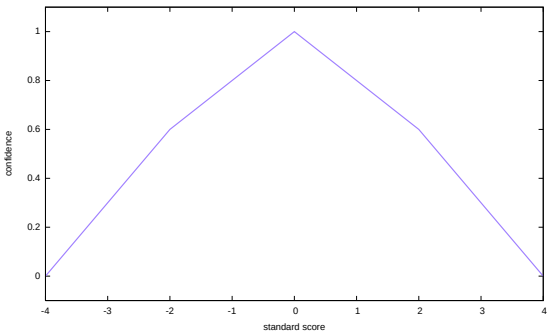
Lacking information as to the precise distribution of each statistic, it would be difficult to map these standard scores directly into a probability. Such an effort would probably be misspent, in any event, because, as noted below, we wish to preserve the flexibility of combining test results in a variety of different ways.

Consequently, the standard scores are mapped onto a confidence value in the range 0.0-1.0 via a piecewise linear function that is metadata field-specific and collection-specific (and that can also be overridden in a validation specification as described below).

The shape of the normalizer function can be tuned to adjust the strictness of the tests and to guarantee the appropriate interpretation of the standard score. For example, the normalizer for the dictionary and vocabulary tests are typically monotonically increasing functions (e.g., Figure 2(a)) because higher-than-average values for these statistics imply increased confidence. On the other hand, the normalizer for the length test would typically peak at the average, as both lower-than-average and higher-than-average values imply lower confidence (Figure 2(b)).



(a) Dictionary and Vocabulary



(b) Length

Figure 2: Typical Normalization Functions

3.4 Validation Specifications and Scripts

The metadata record extracted from a document will typically consist of many fields, which vary both from one collection to another and for different kinds of documents within a collection. Each field may be subject to several different validation tests. The selection of the appropriate tests for a document is therefore a non-trivial activity requiring substantial flexibility, and the combination of a number of separately computed test confidence values into an overall value for each field and for the overall output set calls for considerable flexibility.

```
  ⋮
<val:field name="PersonalAuthor">
   <val:average>
      <val:length/>
      <val:max>
        <val:phrases length="1"/>
<val:phrases length="2"/>
<val:phrases length="3"/>
      </val:max>
   </val:average>
</val:field>
<val:field name="ReportDate">
   <val:reportFormat/>
</val:field>
  ⋮
```

Figure 4: Field Tests Within a Validation Spec. for the DTIC Collection

We have addressed this by creating a special-purpose language (embedded in XML) for describing the selection of tests for a field and the computation of combined confidence values over multiple tests.

Figure 3 shows the high-level structure of a validation specification indicating that the overall confidence for a collection of metadata will be computed as an average of confidences computed for four different metadata fields.

Besides the `average` function (which can be weighted), the specification language permits other combinations including min, max, and weighted sum and also allows rescaling of computed values via arbitrary piecewise linear functions.

Figure 4 shows part of the missing detail from that specification. It indicates that the personal author field is validated using a combination of length and phrase (vocabulary) statistical tests and that the report date field is validated using a deterministic format test.

To validate an extracted metadata record, a simple XML transformation is used to combine each extracted field value with the `<val:field>` structure from the validation specification. The result is a *validation script*, which can be executed.

```
<?xml version="1.0"?>
<metadata>
  <UnclassifiedTitle>Thesis Title: Intrepidity,
     Iron Will, and Intellect: General Robert
     L. Eichelberger and Military Genius
  </UnclassifiedTitle>
  <PersonalAuthor>
     Name of Candidate: Major Matthew H. Fath
  </PersonalAuthor>
  <ReportDate>
    Accepted this 18th day of June 2004 by:
  </ReportDate>
  <approvedby>Approved by:
    Thesis Committee Chair Jack D. Kem, Ph.D.
    , Member Mr. Charles S. Soby, M.B.A.
    , Member Lieutenant Colonel John A. Suprin, M.A.
  </approvedby>
  <acceptedby>
     Robert F. Baumann, Ph.D.
  </acceptedby>
</metadata>
```

Figure 5: Sample Metadata Record (including mistakes)

The semantics of the validation scripting language is implemented in Jelly, a framework for defining executable XML languages[15]. The output of the script is the metadata record (possibly re-ordered) with attributes attached to each field indicating the confidence value for that field and to the record root indicating the overall confidence value for the output as a whole. Figure 6 shows how the metadata record of Figure 5 might have been marked up by the script produced from the specification of Figures 3 and Figures 4. In addition, if a field confidence is less than 0.5, a warning message is attached to the output field. Output records containing such warnings will be referred to human operators for inspection and, if necessary, correction.

```
<val:validate collection="dtic"
  xmlns:val="jelly:edu.odu.cs.dtic...>
 <val:average>
   <val:field name="UnclassifiedTitle">...</val:field>
   <val:field name="PersonalAuthor">...</val:field>
   <val:field name="CorporateAuthor">...</val:field>
   <val:field name="ReportDate">...</val:field>
 </val:average>
</val:validate>
```

Figure 3: High-Level of Validation Spec. for the DTIC Collection

```
<?xml version="1.0"?>
<metadata confidence="0.460"
   warning="ReportDate field does not match required pattern">
  <UnclassifiedTitle confidence="0.979">
     Thesis Title: Intrepidity, Iron
     Will, and Intellect: General Robert
      L. Eichelberger and Military Genius
  </UnclassifiedTitle>
  <PersonalAuthor confidence="0.4"
     warning="PersonalAuthor: unusual number of words">
     Name of Candidate: Major Matthew H. Fath
  </PersonalAuthor>
  <ReportDate confidence="0.0"
     warning="ReportDate field does not match required pattern">
    Accepted this 18th day of June 2004 by:
  </ReportDate>
  <approvedby warning="unvalidated">Approved by:
    Thesis Committee Chair Jack D. Kem, Ph.D.
    , Member Mr. Charles S. Soby, M.B.A.
    , Member Lieutenant Colonel John A. Suprin, M.A.
  </approvedby>
  <acceptedby warning="unvalidated">
     Robert F. Baumann, Ph.D.
  </acceptedby>
</metadata>
```

Figure 6: Sample Output from the Validator

4 Evaluating The Validator

To determine if the validator described here would actually be useful, it was applied to the output of a pre-release version of the metadata extraction system on a number of documents.

Determining whether the validator's confidence values were "reasonable" was somewhat problematic as the exact confidence values could not be predicted or even specified to high precision for realistic outputs. Validation of expert systems, knowledge bases, and other A.I. applications frequently work by comparing the recommendations of the software to that of a human expert, e.g., [9].

In that spirit, we opted to take advantage of the multi-template structure of the extractor. For a group of documents that had been previously assigned to document classes by manual inspection, the template for that manually assigned class and for the largest other classes were all applied to each document. The resulting multiple sets of extracted metadata were then passed through the validator. The purpose of this exercise is to see if the validator's score for the best quality output among the alternatives corresponds to the prior classification performed by human inspection of the documents.

Four templates/document classes were chosen for this experiment. Within the DTIC collection, between half and two thirds of the documents (depending upon whether one looks at the entire collection or only at documents added in recent years) contain a special form in which metadata fields are enumerated and filled in. Such forms are easily recognized and are highly distinctive. As such, it was felt that these would not offer an informative exercise of the validator. Among the remaining documents with no such forms, the metadata extractor must find metadata-bearing pages and locate the metadata field values as instructed by the template. We selected the four most commonly encountered document classes (and their accompanying templates) from a set of relatively recent documents that had previously been downloaded for testing of the extraction system.

The document classes and their templates have been given arbitrary names by the template authors, usually based upon characteristics observed in the first few documents of that class. The four classes used are

au Research reports in the format employed by the Air Command and Staff College, Air University (e.g., Figure 7(a)).

eagle Masters degree theses and similar documents from the U.S. Military Academy at West Point (named for an eagle appearing as a background graphic on the title page) (e.g., Figure 7(b))

rand Research reports and notes from the RAND Corporation (e.g., Figure 7(c))

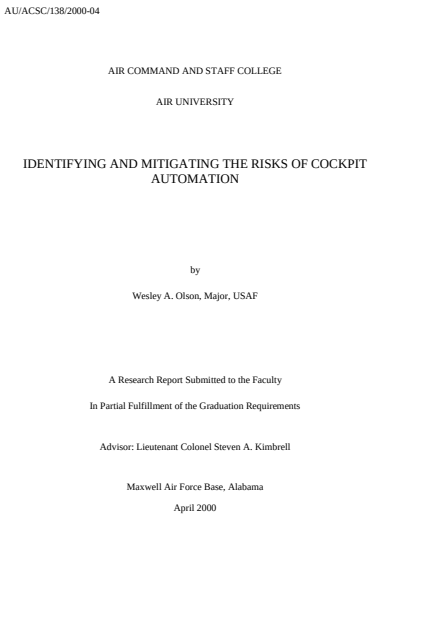
title Another format for RAND Corporation reports, for which the current version of the extractor is able to extract only the document title. (e.g., Figure 7(d))

Table 4 shows the metadata fields extracted by the templates for these document classes.

Figure 8 shows the validation specification intended for use with the DTIC collection. It computes the average confidence in the fields UnclassifiedTitle, PersonalAuthor, CorporateAuthor, and ReportDate. This was judged to be somewhat inappropriate for the evaluation technique proposed for this experiment however. The use of an overall average would unfairly penalize templates that had successfully extracted additional fields at marginally lower confidences. For example, if the rand and title templates each extracted the same title field at a confidence of 1.0, but the rand template also extracted a personal author with confidence 0.95, a simple average over the extracted fields would favor the title template, but human observers would favor the rand template.

An adjustment was therefore made by replacing the overall average by a simple sum, and applying a rescaling to each field so that confidence values of less than 0.5 were rescaled to small negative values, so that fields extracted with unacceptably low confidence would not be treated as positive factors in the overall sum. Figure 9 illustrates these changes.

Table 5 shows the results of applying the adjusted specification to the template outputs for 167 documents. For approximately 74% of these documents, the validator gave the highest confidence score to the



AU/ACSC/138/2000-04

AIR COMMAND AND STAFF COLLEGE

AIR UNIVERSITY

IDENTIFYING AND MITIGATING THE RISKS OF COCKPIT AUTOMATION

by

Wesley A. Olson, Major, USAF

A Research Report Submitted to the Faculty

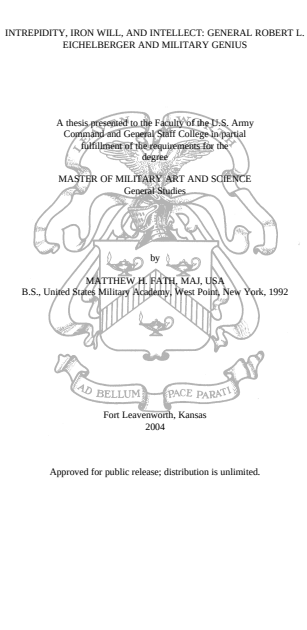
In Partial Fulfillment of the Graduation Requirements

Advisor: Lieutenant Colonel Steven A. Kimbrell

Maxwell Air Force Base, Alabama

April 2000

(a) au



INTREPIDITY, IRON WILL, AND INTELLECT: GENERAL ROBERT L. EICHELBERGER AND MILITARY GENIUS

A thesis presented to the Faculty of the U.S. Army Command and General Staff College in partial fulfillment of the requirements for the degree

MASTER OF MILITARY ART AND SCIENCE
General Studies

by

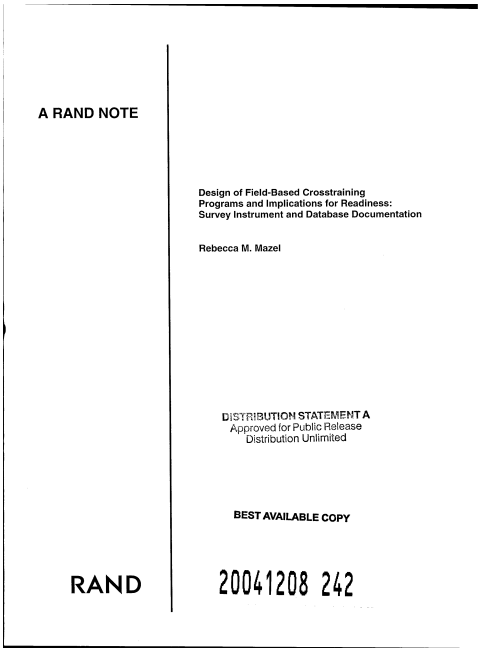
MATTHEW H. FATH, MAJ, USA
B.S., United States Military Academy, West Point, New York, 1992

AD BELLUM PACE PARATI

Fort Leavenworth, Kansas
2004

Approved for public release; distribution is unlimited.

(b) eagle



A RAND NOTE

Design of Field-Based Crosstraining Programs and Implications for Readiness: Survey Instrument and Database Documentation

Rebecca M. Mazel

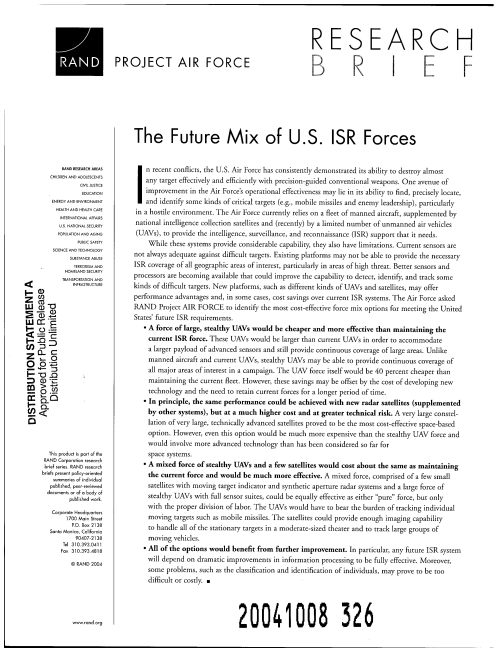
DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

BEST AVAILABLE COPY

RAND

20041208 242

(c) rand



RAND PROJECT AIR FORCE

RESEARCH BRIEF

The Future Mix of U.S. ISR Forces

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

In recent conflicts, the U.S. Air Force has consistently demonstrated its ability to destroy almost any target effectively and efficiently with precision-guided conventional weapons. One avenue of improvement in the Air Force's operational effectiveness may lie in its ability to find, precisely locate, and identify some kinds of critical targets (e.g., mobile missiles and enemy leadership), particularly in a hostile environment. The Air Force currently relies on a fleet of manned aircraft, supplemented by national intelligence collection satellites and (recently) by a limited number of unmanned air vehicles (UAVs), to provide the intelligence, surveillance, and reconnaissance (ISR) support that it needs.

While these systems provide considerable capability, they also have limitations. Current sensors are not always adequate against difficult targets. Existing platforms may not be able to provide the necessary ISR coverage of all geographic areas of interest, particularly in areas of high threat. Better sensors and processors are becoming available that could improve the capability to detect, identify, and track some kinds of difficult targets. New platforms, such as different kinds of UAVs and satellites, may offer performance advantages and, in some cases, cost savings over current ISR systems. The Air Force asked RAND Project AIR FORCE to identify the most cost-effective force mix options for meeting the United States' future ISR requirements.

- **A force of large, stealthy UAVs would be cheaper and more effective than maintaining the current ISR force.** These UAVs would be larger than current UAVs in order to accommodate a larger payload of advanced sensors and still provide continuous coverage of large areas. Unlike manned aircraft and current UAVs, stealthy UAVs may be able to provide continuous coverage of all major areas of interest in a campaign. The UAV force itself would be 40 percent cheaper than maintaining the current fleet. However, these savings may be offset by the cost of developing new technology and the need to retain current forces for a longer period of time.
- **In principle, the same performance could be achieved with new radar satellites (supplemented by other systems), but at a much higher cost and at greater technical risk.** A very large constellation of very large, technically advanced satellites proved to be the most cost-effective space-based option. However, even this option would be much more expensive than the stealthy UAV force and would involve more advanced technology than has been considered so far for space systems.
- **A mixed force of stealthy UAVs and a few satellites would cost about the same as maintaining the current force and would be much more effective.** A mixed force, comprised of a few small satellites with moving target indicator and synthetic aperture radar systems and a large force of stealthy UAVs with full sensor suites, could be equally effective as either "pure" force, but only with the proper division of labor. The UAVs would have to bear the burden of tracking individual moving targets such as mobile missiles. The satellites could provide enough imaging capability to handle all of the stationary targets in a moderate-sized theater and to track large groups of moving vehicles.
- **All of the options would benefit from further improvement.** In particular, any future ISR system will depend on dramatic improvements in information processing to be fully effective. Moreover, some problems, such as the classification and identification of individuals, may prove to be too difficult or costly. ■

This product is part of the RAND Corporation research brief series. RAND research briefs present policy-oriented summaries of individual published, peer-reviewed documents or of a body of published work.

Corporate Headquarters
1700 Main Street
P.O. Box 2138
Santa Monica, California
90407-2138
Tel 310.393.0411
Fax 310.393.4818

© RAND 2004

www.rand.org

20041008 326

(d) title

10

Figure 7: Samples of Document Formats Employed

Field	Document Class			
	au	**eagle**	**rand**	**title**
UnclassifiedTitle	·	·	·	·
PersonalAuthor	·	·	·	
CorporateAuthor	·			
ReportDate	·	·		
Identifier	·			
Advisor	·			
ApprovedBy		·		
AcceptedBy		·		

Table 4: Fields Extracted by Each Template

```
<?xml version="1.0"?>
<val:validate collection="dtic"
 xmlns:val="jelly:edu.odu.cs.dtic.validation.ValidationTagLibrary">
 <val:sum>
   <val:field name="UnclassifiedTitle">
      <val:rescale function="0.499 -0.01 0.5 0.5 1.0 1.0">
        <val:average>
           <val:dictionary/>
           <val:length/>
        </val:average>
      </val:rescale>
   </val:field>
   <val:field name="PersonalAuthor">
      <val:rescale function="0.499 -0.01 0.5 0.5 1.0 1.0">
         ⋮
```

Figure 9: Adjusted Validation Specification for This Experiment

Manually Assigned Class	**Number of documents Validator Preferred**				**Total Documents**
	Au	**Eagle**	**Rand**	**Title**	
Au	86	0	0	0	86
Eagle	0	8	33	4	45
Rand	0	0	8	4	12
Title	0	0	1	23	24
				Total:	167

Table 5: Validator's Choices (Highest Confidence) versus Human's

```
<?xml version="1.0"?>
<val:validate collection="dtic"
 xmlns:val="jelly:edu.odu.cs.dtic...>
   <val:average>
     <val:field name="UnclassifiedTitle">
<val:average>
   <val:dictionary/>
   <val:length/>
</val:average>
     </val:field>
     <val:field name="PersonalAuthor">
       <val:min>
  <val:length/>
  <val:max>
     <val:phrases length="1"/>
     <val:phrases length="2"/>
     <val:phrases length="3"/>
  </val:max>
       </val:min>
     </val:field>
     <val:field name="CorporateAuthor">
<val:min>
   <val:length/>
   <val:max>
      <val:phrases length="1"/>
      <val:phrases length="2"/>
      <val:phrases length="3"/>
      <val:phrases length="4"/>
   </val:max>
</val:min>
     </val:field>
     <val:field name="ReportDate">
<val:dateFormat/>
    </val:field>
  </val:average>
</val:validate>
```

Figure 8: Validation Specification for DTIC Collection

same document template as had been anticipated by the human evaluator.

Most of the differences in the highest confidence assigned by the validator as opposed to the visual assignment made by a human were associated with documents the human had placed into the eagle class. Inspection of these cases found a common set of problems in the metadata extracted by the template for that class. These are illustrated in the output metadata shown earlier in Figure 5, which was extracted by the eagle template. The title includes the inappropriate lead-in phrase "Thesis title':" - minor defect that causes many titles in the class to score slightly lower than the titles extracted by other templates because they contribute to slightly unusual length. More problematic is the phrase "Name of Candidate:" and the military ranks appearing the the PersonalAuthor field. These are scored poorly on the basis of inappropriate vocabulary (the phrase tests) and abnormal length. Finally, this template extracts report dates that, again, include a great deal of extraneous text.

So, in these cases, it appears that the validator did exactly what it was intended to do, flagging field values that we would not wish to pass into a database without intermediate clean-up. The metadata extractor design, in fact, calls for a as-yet-unimplemented post-processing stage intended to clean up these kinds of problems. When the validator was rerun on a set of metadata where the action of this post-processor had been simulated by simple editing scripts, all 85 documents assigned by the human to the eagle class had that template preferred by the validator.

There are 5 remaining cases where the validator preferred a template other than one one visually assigned by a human. Four of these are in the rand class. Inspection of these cases showed that, in each of these four cases, the rand and title templates extracted exactly the same title, but the rand template claimed to have extracted an author name that was, in fact incorrect. In two instances, the rand template mistakenly identified a public release waiver as the author name. In the other two instances, OCR errors led to strings of garbled characters in the location where the author name should have been. In

each case, the validator preferred the title template rather than the combination of the same title with a poorly extracted author.

In only one instance did the validator prefer a result that a human would not have selected. That instance was traced to a defect in the design of the phrase tests, which return an unsuitably high confidence value in cases where no phrases of the desired length can exist (i.e., a test on phrases of length three when the extracted metadata consists of only two words).

The net result, then is that in more than 99% of the documents examined, the validator made judgments compatible with those that a human inspector of the output would have made.

5 Conclusions

We have described a system for dynamic validation of a system for extracting metadata from scanned documents. Because this is not an application where provision of an exact oracle is possible, we have employed a series of statistical models of output characteristics.

Flexibility and adaptability of the validator was achieved by developing a customized scripting language describing how the various tests should be combined to obtain an overall measure of confidence in a program output.

We believe that a similar statistical dynamic validation process could be useful in a variety of applications where the specification of exact oracles is impossible or impractical. Although this application may have been unusually fortunate in the amount of prior good-quality outputs available, in other circumstances the statistics could be accumulated during the early stages of deployed operations by starting with a high degree of human inspection of program outputs, then relaxing the degree of direct human oversight as the body of human-approved outputs is accumulated.

The suitability of the validator was demonstrated by an experiment measuring its ability to mimic human judgments as to which of several alternate outputs for the same document would be preferred. It performed better than had actually been anticipated, agreeing with human judgments in more than 99% of the documents examined.

Curiously, this is a more consistent performance than we had been able to obtain from the various vision-inspired algorithms for classifying document layouts prior to performing the actual metadata extraction. We are currently considering the possibility of exploiting the validator as a kind of "post hoc" classification, applying all available templates to each document and selecting the one that scores best under validation.

6 Acknowledgments

This work was supported by grants from the Defense Technical Information Center and the National Aeronautics and Space Administration.

References

[1] W. T. Chen, J. P. Ho, and C. H. Wen. Dynamic validation of programs using assertion checking facilities. In *Proc. COMPSAC 78*, pages 533–538, Nov. 1978.

[2] L. A. Clarke and D. S. Rosenblum. A historical perspective on runtime assertion checking in software development. *SIGSOFT Softw. Eng. Notes*, 31(3):25–37, 2006.

[3] D. F. McAllister and M. A. Vouk. Fault-Tolerant Software Reliability Engineering. In Michael R. Lyu, editor, *Handbook of Software Reliability Engineering*. McGraw Hill, New York, NY, USA, 1996.

[4] M. D. Davis and E. J. Weyuker. Pseudo-oracles for non-testable programs. In *ACM 81: Proceedings of the ACM '81 conference*, pages 254–257, New York, NY, USA, 1981. ACM Press.

[5] A. K. Deb and A. L. Goel. Model for execution time behavior of a recovery block. In *Proceedings COMPSAC 86*, pages 497–502, 1986.

[6] Defense Technical Information Center. Public Scientific and Technical Information Network. `http://stinet.dtic.mil/str/index.html`, 2007.

[7] R. S. Fabry. Dynamic verification of operating system decisions. *Commun. ACM*, 16(11):659–668, 1973.

[8] J. Grundy, G. Ding, and J. Hosking. Deployed software component testing using dynamic validation agents. *J. Syst. Softw.*, 74(1):5–14, 2005.

[9] C. Hernandez, J. J. Sancho, M. A. Belmonte, C. Sierra, and F. Sanz. Validation of the medical expert system renoir. *Comput. Biomed. Res.*, 27(6):456–471, 1994.

[10] National Aeronautics and Space Administration. NASA Technical Reports Server. `http://ntrs.nasa.gov/search.jsp`, 2007.

[11] B. Randell. System structure for software fault tolerance. *SIGPLAN Not.*, 10(6):437–449, 1975.

[12] D. S. Rosenblum. A practical approach to programming with assertions. *IEEE Trans. Softw. Eng.*, 21(1):19–31, 1995.

[13] J. Tang. *Template-based Metadata Extraction for Heterogeneous Collections*. PhD thesis, Old Dominion University, 2006.

[14] J. Tang, K. Maly, S. Zeil, and M. Zubair. Automated Building of OAI Compliant Repository from Legacy Collection. In *Proceedings of the 10th International Conference on Electronic Publishing (ELPUB)*, June 2006.

[15] The Jakarta Project (Apache). Jelly: Executable XML. `http://jakarta.apache.org/commons/jelly/`, 2006.

[16] U.S. Government Printing Office. A Strategic Vision for the 21st Century. Technical report, 2004.

[17] E. J. Weyuker. On testing nontestable programs. *The Computer Journal*, 25(4), 1982.

[18] S. S. Yau and R. C. Cheung. Design of self-checking software. In *Proceedings of the international conference on Reliable software*, pages 450–455, New York, NY, USA, 1975. ACM Press.