Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Linguistic Text Mining for Problem Reports

Jane T. Malin
Software, Robotics and Simulation Division
NASA Johnson Space Center
Houston, USA
jane.t.malin@nasa.gov

David R. Throop
The Boeing Company
Houston, USA
david.r.throop@nasa.gov

Christopher Millward, Hansen A. Schwarz and Fernando Gomez
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, USA
cmillward@cs.ucf.edu, hschwartz@cs.ucf.edu, gomez@eecs.ucf.edu

Carroll Thronesbery
S&K Aerospace, Incorporated
Houston, USA
carroll.g.thronesbery@nasa.gov

***Abstract*—This paper describes a linguistic text mining tool for analyzing problem reports in aerospace engineering and safety organizations. The Semantic Trend Analysis Tool (STAT) helps analysts find and review recurrences, similarities and trends in problem reports. The tool is being used to analyze engineering discrepancy reports at NASA Johnson Space Center. The tool has been augmented with a statistical natural language parser that also resolves parsing gaps and identifies verb arguments and adjuncts. The tool uses an aerospace ontology augmented with features of taxonomies and thesauruses. The ontology defines hierarchies of problem types, equipment types and function types. STAT uses the output of the parser and the aerospace ontology to identify words and phrases in problem report descriptions that refer to types of hazards, equipment damage, performance deviations or functional impairments. Tool performance has been evaluated on 120 problem descriptions from problem reports, with encouraging results.**

***Keywords*—text mining, natural language understanding, knowledge discovery, ontology**

I. Introduction

Problem reports and change requests are used to document, track and analyze problems and associated investigations, dispositions, corrective actions and outcomes. Analysts look for recurrences and trends in problem reports. Analyzing groups of similar problems helps identify causes and corrective actions with wide impacts. Identifying past problems that are similar to a new problem can also help organizations handle the new problem.

There are many ways that problem reports can be similar. It can be easy to overlook recurrences and trends. For this reason, problem reports are frequently stored as multiple-field records in databases. The problem records include multiple codes that are designed to help classify and group the reports. However, the codes can be confusing or out of date, and they may not match the problem situation being reported. Key information about a problem is commonly embedded in free text fields that contain short titles or text summaries. Individual keywords can be extracted from the texts, but simple keyword approaches are as brittle as codes. There are too many ways of conveying an idea in natural language.

Text and data mining approaches have been developed for natural language to overcome the difficulties with codes, keywords and search. When successful, these mining approaches help analysts discover new recurrences and trends. Similar problem reports can be grouped by measuring statistics of text co-occurrences [1]. To apply the approach, very little needs to be known about the texts. However, it can be difficult and frustrating for analysts to find out whether the similarity between problem reports in a cluster is meaningful. Meaningful clusters can be missed when syntactically equivalent words and phrases cannot be recognized.

This paper describes the Semantic Trend Analysis Tool (STAT), which uses a linguistic text mining approach to overcome difficulties with both codes and statistical text mining. This approach uses an Aerospace Ontology (AO) to interpret categorize and tag key information in the text. AO was developed to classify problems (hazard, damage, impairment or discrepancy) and the things that have the problems (objects, actions and events) [2]. The first version of the parser extracted key information by identifying and parsing noun phrases, stemming to convert to verb forms, and capturing modifying style words [3]. The new version is enhanced with a sophisticated parser.

The following sections of the paper discuss the analyst tasks supported by STAT, the ontology and the parsing and tagging approaches. Finally, there is an evaluation of the performance improvements in STAT resulting from enhanced linguistic analysis.

II. Analyzing Tagged Problem Reports

STAT helps problem report analysts discover related groups of problem reports and trends in their frequencies over

The authors thank Robert Beil, Dawn Schaible and the NASA Engineering and Safety Center for supporting this work.

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

time. STAT has been applied to engineering discrepancy reports (DRs) at NASA Johnson Space Center.

The analyst is responsible for discovering problem groups and trends so that corrective actions can be formulated. For example, if a single battery has expired, there is probably no corrective action beyond replacing the battery. However, if there is a growing trend of expired batteries, then practices for ordering and storing them may need to be changed.

STAT downloads DR records from a database repository. It is typical to download a year of reports, grouped by quarter, to assess the development of trends. STAT performs linguistic analysis on the problem description field in each report record. It tags the records according to equipment types and problem types mentioned in the text. The results of these automated analyses are presented in an integrated collection of Web pages. The analyst uses the Web site to gain an overview of the collection of problem reports, to discover groups of problem reports, and to identify trends for problem types and equipment types. This helps the analyst focus the investigation on interesting groups of problems.

A. *Web Site for Analysts*

The overview Web page contains summary charts and graphs. They show the number of problem reports for each quarter, broken out by a number of variables important to DR analysts. For example, there are summaries for the number of reports assigned to each analysis team, for each responsible organization, for equipment types, and for problem types. These summary views allow the analyst to gain familiarity with where the most frequent problems have occurred and whether they have increased over the course of the year. The problem types and equipment types correspond to the tags assigned to the problem reports by STAT. Fig. 1 shows an example summary graph for problem types by quarter.

These summaries and trend reports help the analyst to vet the quality of the problem report batch and to gain confidence that problems are being adequately analyzed by STAT.

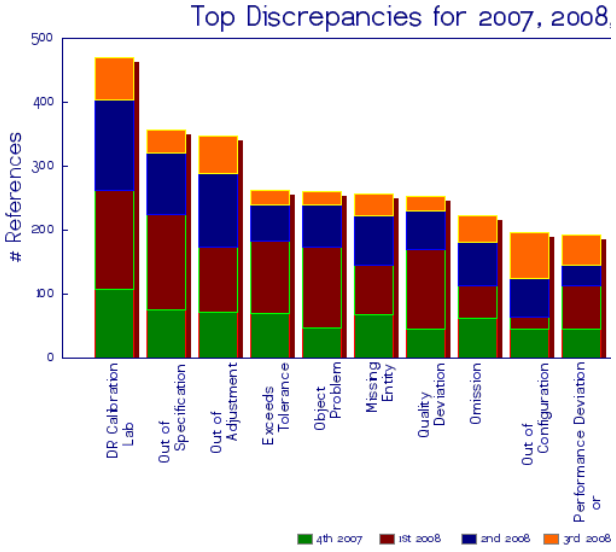

Figure 1. A summary of top problem types by quarter.

B. *Discovery: Hierarchically Grouped Problem Reports*

Having gained an overview of the last year of DRs, the analyst can discover prevalent problem groups. For example, the analyst might be interested in sharp edges, especially those that could threaten the integrity of astronaut gloves. A link from the overview page leads to a hierarchical table display of AO problem types, as shown in Fig. 2. The table indicates frequencies of numerous problem reports that use "sharpness" problem words, including 47 problem reports that mention sharp edges. The table also allows the analyst to navigate to specialized Web pages with more information on each of these problem groups.

C. *Trends for Problem Types and Equipment Types*

Fig. 3 illustrates a graph on the Web page for sharpness problems. This graph shows types of hardware by quarter, using a hardware type code from the database records. The analyst is concerned about sharp edges on flight hardware. It appears that almost all of the DRs about sharpness concerned flight hardware (blue portion of each bar). Sharpness problems decrease after the third quarter. This suggests that measures taken in the third quarter are beginning to be effective.

To form a complete interpretation of these numbers, the analyst compares these summaries to knowledge of discrepancies submitted during the last year and to knowledge of related events. For example, if the analyst had external evidence that there was a flurry of sharpness problems in the last quarter, the difference between external evidence and this report would be investigated.

2.4.1.5.5	Sharpness		
	CUTTING	12	3V0730029 KM EF0830263 1W
	SHARP	1	E10730171
	POINT	1	310830001
	ABRASIVE	1	E60830018
	SHARP EDGE	47	KM0730012 E1 2V0830006 7J EF0830160 E60 180830028 180 180830034 EB0 180830041 EF0

Figure 2. A hierarchical grouping of problem reports.

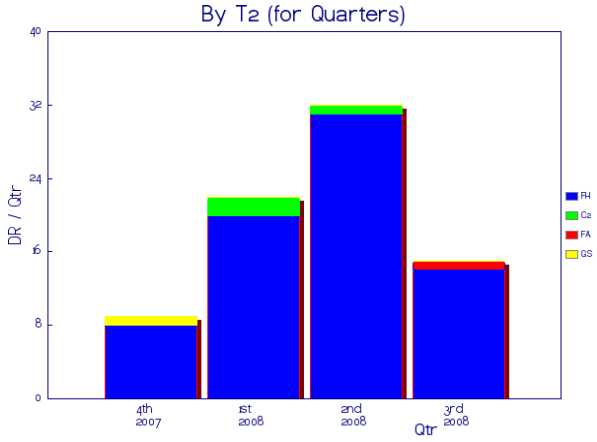

Figure 3. A closer look at sharpness for flight hardware.

D. *Additional Support for Analysts*

The Web site links to individual DRs in the database, as shown in Fig. 4. Each record contains a number of Trend Codes (T2-T10), which were designed to aid trend analysis. These codes were not successful until combined with the tags produced by STAT. Individual records help the analyst gain more knowledge of the reports that compose a particular grouping. With this information, the analyst can construct accurate descriptions of problem groups, trends and possible corrective actions.

After gaining an understanding of where most of the problems have occurred, analysts can further study the problem reports in the group to determine possible underlying causes and corrective actions.

III. AEROSPACE ONTOLOGY FOR TAGGING

Linguistic parsers can handle many difficulties with interpreting and tagging text in problem reports. However, there are some serious remaining problems that result from using nonstandard or specialized terminology that is common in technical descriptions. It is important to handle synonyms, uncommon spellings, acronyms and terms that are not synonymous but are closely related. The classification hierarchies and mapping words and phrases in the AO are designed to directly address these semantic problems.

Each AO class is situated in a classification hierarchy that can be used to identify related classes—super-classes, subclasses and sibling classes. Each AO class includes a list of "mapping" words and phrases that help identify terminology that is typically used to describe that type of thing. These can include synonyms, common misspellings and acronyms from the problem report text corpus. These mapping words and phrases give AO some thesaurus features. For example, several mapping words for sharpness are shown in Fig. 2.

STAT finds matches between AO mapping words or phrases and the parsed nouns, verbs and associated modifiers. These matches are used to identify the types of problems and types of objects in the text. One or more problem classes and one or more object classes can be associated with each problem report text field. These identified classes are added to the problem report record as tags.

Generally, problems can be described as objects or occurrences that have some bad property. Thus, AO uses hierarchies of types of properties and values (types of influence, expense, goodness, authorization, effectiveness, definiteness and soundness).

DocumentNumber: EF0830062	**ClosedDate**: 3/10/2008	**InitiationDate**: 2/7/2008	**RecordCenter**: CTS3
ResponsibleOrganization: EC	**FinalDisposition**: REPAIR	**ProjectCode**: 00049	**SerialNumber**: 1002
T2: FH	**T3**: PC	**T4**: FX	**T5**: DS1
T6: AE	**T7**: CL	**T8**: RA	**T9**: JS
T10: 11			
PartName: OCK BRACKET2, POP-UP; ALODINE			
DocumentTitle: . IT WAS NOTICED DURING RECEIVING INSPECTION, THAT THELOCK BRACKET2, POP-UP, HAS A BURR/SHARP EDGE ON 1 OF 2, .213 X .313 SLOTS. LOCATED AT THE .500 LOCATION, ZONE C7.			

Figure 4. A view of the individual discrepancy report record.

These properties are combined with objects (types of energy, life forms, devices, structures, substances, collections) and occurrences (types of events, processes, actions, operations), to define problem types and mapping phrases for types of problems.

AO includes a wide variety of types of problems described in natural language, including hardware problems, human and organizational problems and software problems [cf. 2]. Problems, failures, mishaps and accidents can be classified as types of damage and impairments, hazardous sources of damage or impairment, and performance deviations or errors. There are currently approximately 350 classes in the Problem Hierarchy. Fig. 5 shows an example of a small part of AO, showing hierarchical class structure. The figure shows the mapping words for the sharpness problem type.

The AO hierarchies do not form a strict taxonomy. Combinations of distinctions or facets can be used to form a class, and a class can have multiple parent classes. Assignment to classes is not mutually exclusive since many words have multiple meanings. Thus, mapping words can be associated with multiple classes. This flexibility is needed so that multiple groupings of problem reports can be derived for multifaceted problem descriptions.

IV. LINGUISTIC TEXT MINING FOR PROBLEM REPORTS

The first successful STAT implementation was based on the Reconciler parser and tagger. The parser breaks each sentence into clauses (*subject, verb, object, prepositional-phrase…)*. To tag a phrase, the tagger searches for concepts (*ALIGN, CLEARANCE)* and style words. In English, many phrases that connote problems consist of (1) a desirable quantity, along with (2) a negation or violation of that quantity (e.g., *not sufficient clearance, badly aligned, none available).* The negation or violation words are called "styles."

Mapping phrases from the ontology are stored as combinations of styles and concepts. If a concept is modified by a style, and if the style-concept pair is indexed as a problem type, then the phrase is tagged. Some words in the ontology are problem types without any style. These should not be tagged if they *are* styled – *not corroded* should not be tagged as *CORRODED.*

Mechanical Burden

- Stress or Load
- Friction Burden
- Abrasion
- Vibration Burden (mechanical)
  - Spasms
- Sharpness: cutting, sharp, pointed, keen, sharp edge, sharp corner, snag, jagged, protruding
- Acceleration
  - Falling
  - Erratic Motion
    - Stumbling

Figure 5. Small hierarchy of Mechanical Burden problem types.

When the tagger finds a style word and a concept, it must decide the style scope—whether the style word modifies the concept. Scope is determined heuristically. If style words and concepts co-occur in a clause, the style is judged to modify the concept. This heuristic does a very good job of tagging when the styles are prefixes (*misalignment)* or when words and the concepts are adjacent (the most common case). The heuristic breaks down when: 1) styles and concepts are not adjacent; 2) there are conjuncts (e.g., *the door had inadequate paint and good clearance);* or 3) a noun-concept is not the head of a noun-phrase (e.g., *it passed the insufficient-clearance test)*.

Mistakes in determining the scope of style words can result in false positives or false negatives in the tagging. While the first STAT parser produced better results for analysts than using Trend codes, a new parsing algorithm could solve the scope problems. The next two sections describe the new parser and a before-after evaluation of STAT tagging performance.

V. NEW PARSING ALGORITHM

At the heart of the parsing system is a syntax-based algorithm for identifying the arguments and adjuncts of the verbs from constituent parse trees. The term "verb scope" is used to refer to both the arguments and the adjuncts of a verb [4]. However, the scope of a verb does not indicate which constituents are adjuncts and which are arguments. The algorithm is based on the output of the Charniak parser [5], which is a state-of-the-art constituent parser.

The algorithm focuses on recovering gaps and resolving empty nodes. In a typical constituent parse tree, each phrase in a sentence occurs only once in the parse tree, even though in the argument structure it may occur in multiple places. Consider the sentence, "After being scrutinized by the newspaper the politicians demanded an apology." While "the politicians" occurs only once in the sentence, it is both an object of "scrutinized" and the subject of "demanded." In Fig. 6, the Charniak parse tree of this sentence has only one occurrence of the "the politicians" whereas in the transformed parse tree an empty node referring to "the politicians" has been inserted before "scrutinized" to resolve the understood structure of the sentence. By resolving these constructions, the algorithm is able identify the arguments and adjuncts of the main verbs from such parse trees.

The scoping algorithm runs on a parse of a sentence, from which it produces a list of scopes for each of the main verbs in the sentence. The scoping algorithm is done in two parts. The first part is a cascade of transformations on the parse tree itself to annotate complex constructions and resolve gaps. The second part then operates on the modified parse tree and builds the scopes, attempting to resolve the gaps left unresolved from the first step. Fig. 6 shows an example of a modified parse tree and the scopes of its verbs. The use of patterns allows us to deal with parse trees at a sufficient level of abstraction. These patterns are a form of regular tree expression and can include the distinctive features of constituents, which are tag, annotation, word (if applicable) and positional information.

A. Tree Transformations

The tree transformation algorithm works by applying general rules to the parse tree. The rules recognize known syntax patterns and either annotate nodes to recognize complex constructions or resolve empty nodes that reference other nodes in the parse tree. The tree transformations first walk pre-order through the tree and apply each transformation to every node of the tree before moving to the next transformation. This assures us that look-ahead matching will not fail because a construct wasn't identified despite being present in the tree. The tree transformation algorithm first annotates constructions such as reduced clausal modifiers, verb coordination and relative clauses. Then, the algorithm inserts empty nodes into the parse tree to resolve gaps for known resolvable constructions.

B. Scope-building Algorithm

Once the parse tree has been modified and tagged, the annotated tree is then processed and a set of scopes is constructed for the tree. The algorithm determines the voice of the verb, arguments and adjuncts using syntactic patterns and rules. In cases where the subject cannot be determined without semantic analysis, multiple subjects are supplied to allow for differentiation later by a semantic interpreter [6].

Charniak parse tree

```
(S1
 (S
  (PP (IN After)
   (S
    (VP (AUXG being)
     (VP (VBN scrutinized)
      (PP (IN by) (NP (DT the) (NN newspaper)))))))
  (NP (DT the) (NNS politicians))
  (VP (VBD demanded) (NP (DT an) (NN apology)))
  (...)))
```

Transformed parse tree

```
(S1
 (S
  (PP (IN After)
   (S
    (NP-GAP (DT the) (NNS politicians))
    (VP (AUXG being)
     (VP (VBN scrutinized)
      (PP (IN by) (NP (DT the) (NN newspaper)))))))
  (NP (DT the) (NNS politicians))
  (VP (VBD demanded) (NP (DT an) (NN apology)))
  (...)))
```

Scopes for the verbs

: v1
VERB: (VBN scrutinized)
SUBJECT: (NP the politicians)
MODIFIERS: (AUXG being)
PREP-PHRASES: (PP by the newspaper)
VOICE: PASSIVE
POTENTIAL-SUBJECT: (NP the newspaper)

: v2
VERB: (VBD demanded)
SUBJECT: (NP the politicians)
OBJECTS: (NP an apology)
PREP-PHRASES: (PP After (clause: v1))
VOICE: ACTIVE

Figure 6. Transformed parse tree and verb scopes for the sentence, "After being scrutinized by the newspaper the politicians demanded an apology."

When the scoping algorithm fails, it fails gracefully by providing enough information for the next step without having to re-examine the parse tree. In Fig. 7 the initial prepositional phrase "DURING THE FABRICATION OF ONE OF EIGHT PALLET ASSY…" is recovered as being attached to the verb "found." Further, the voice of the clause headed by "found" was determined to be passive, so the algorithm knows that "it," the constituent in the subject position, will be the direct object of the verb, not the subject.

VI. Application of Algorithm

The scoping algorithm is general and can be applied to any English language domain. Around this system is a design that uses the scoping algorithm as a base. The overall system is tuned to the corpus of DRs to produce the type of information required by the STAT tagger.

A. Corpus Pre-processing

The corpus of DRs is very different in content and style from the Wall Street Journal (WSJ) corpus on which the Charniak parser is trained. The approach to acquiring the best parses from the corpus is to pre-process the text, using the Charniak parser as is, without retraining on a different corpus. Thus, the pre-processing can be tuned for different corpora, based on a semantic analysis, without needing an extensive corpus to retrain the Charniak parser. Despite the large differences between the application corpus and the WSJ corpus, the Charniak parser works remarkably well.

Modified Charniak parse tree

```
(S1
 (S
  (PP (IN DURING)
   (NP (NP (DT THE) (NN FABRICATION))
    (PP (IN OF)
     (NP (NP (CD ONE))
      (PP (IN OF)
       (NP (CD EIGHT) (NN PALLET) (NN ASSY)))))))
  (NP (PRP IT))
  (VP (AUX WAS)
   (VP (VBN FOUND)
    (SBAR (IN THAT)
     (S (NP (DT THE) (NN COVER) (NN MATERIAL))
      (VP (AUXVB HAS)
       (NP (CD 3) (JJ MINOR)
           (NN OIL) (NNS STAINS)))))))
  (PERIOD .)))
```

Scopes for the verbs

```
:v1
VERB: (VBN FOUND)
SUBJECT: (NP IT)
MODIFIERS: (AUX WAS)
OBJECTS: (SBAR (clause :v2))
PREP-PHRASES: (PP DURING THE FABRICATION OF
                  ONE OF EIGHT PALLET ASSY)
VOICE: PASSIVE

:v2
VERB: (AUXVB HAS)
SUBJECT: (NP THE COVER MATERIAL)
OBJECTS: (NP 3 MINOR OIL STAINS)
VOICE: ACTIVE
```

Figure 7. Transformed parse tree and verb scopes for the sentence, "DURING THE FABRICATION OF ONE OF EIGHT PALLET ASSY IT WAS FOUND THAT THE COVER MATERIAL HAS 3 MINOR OIL STAINS."

The corpus of reports is all in uppercase. The sentences are often incomplete. Some have typos, many lists, part numbers, poor punctuation and aerospace domain jargon. The wording is often terse. Personal pronouns are omitted. The corpus also has a high frequency of sentence fragments. However, most of the sentences are not overly complex. The majority have one or two clauses. Through corpus specific pre-processing of the text, the Charniak parser can be used with good results.

Each problem description is passed to the system as a single string of text that the system must split into sentences before it proceeds. A set of regular expressions is used to split at the sentence boundaries naively. The stream of "sentences" and punctuation are then algorithmically analyzed to determine the actual boundaries. Common sentence fragments and meta-statements, like "REQUESTED BY..." and due dates are pruned from the sentences that are sent to the parser.

Besides these meta-statements, another problematic pattern is a long list of labels and numbers. Instances like the following cause the parser to incorrectly parse the sentence, or even fail entirely:

> ZONE D7 , .506 +/- .004 , ZONE B7 .378 + .006 - .000 , ZONE D7 2X .876 , ZONE D7 2X .748 , ZONE B7 .145+ .007 - .000 ZONE B7 .200 + .005 - .000 ...

Processing is done to identify long lists by first replacing these with a marker item, then substituting the list fragment back into the parse tree so that the parseable part of the sentence can be handled. The pre-processing is fully programmatic; there is no human intervention.

Despite the irregularities in the DR corpus, the Charniak parser performs well, overall. The parser does worse as sentence complexity increases. Even though the sentences may have odd lists and other fragments, the pruned sentence structure is rather plain. The scoping algorithm is robust, and it works well as long as the Charniak parser produces good parse trees. The scoping algorithm used in conjunction with the analysis and pre-processing of the text from DRs provides accurate verb scopes for STAT tagging.

VII. Evaluating Parsing Improvements

Two test sets were prepared to compare STAT performance before and after integrating the augmented parser. The first set of 60 sentences was constructed to target difficulties for the tagging heuristic. All sentences contained variations on the concept *insufficient clearance,* but with different complicating features—conjunctions, intervening adverbs, odd word order, intervening parenthetical expressions and more. The second test set of 60 representative problem description sentences was selected systematically from the Fiscal Year 2006 set of NASA DRs. Every 60th problem report was selected, while discarding those with problem descriptions that (nearly) duplicated others, were not well formed, were over 150 characters long or in which the analyst could find no problem described. An analyst hand-tagged the problems mentioned in each description.

Both sets of sentences were tagged twice by STAT, using each parser. The STAT-generated tags were compared to the manual tags and scored as true-positive (t_p), false-positive (f_p) and false-negative (f_n). Performance accuracy was measured

for precision (proportion of tags that are good) as in (1) and recall (proportion of good cases that are tagged properly) as in (2). Performance by the two parser versions was compared on each of the two test data sets.

$$Precision = \frac{t_p}{t_p + f_p} \quad (1)$$

$$Recall = \frac{t_p}{t_p + f_n} \quad (2)$$

VIII. Evaluation Results

For both data sets, using the new parsing algorithm substantially improved both precision and recall, as shown in Table 1. Scores can be improved either by correctly tagging occurrences that were missed (increasing tp and decreasing f_n) or by avoiding incorrect tagging (decreasing f_p). In the Insufficient Clearance data set, the improvement was split about evenly between these ways. In the representative sample of DRs, the improvement came largely from decreases in fp. In this data set, the most common improvements were in places where the Reconciler parser scoped a negation term too broadly and the new parsing algorithm constrained it. The lower recall scores in the Insufficient Clearance data set were due to the focus on difficult-to-scope sentences, regardless of their prevalence in natural text.

IX. Discussion and Conclusions

This paper describes the initial integration and evaluation of the new parsing algorithm for STAT. These performance improvements will help clean up the problem groups so that there are fewer false alarms and more hits. STAT performance with the old parser was sufficient to substantially reduce analyst effort. These improvements will increase STAT credibility and reduce analyst work needed to remove false alarms (problem reports that do not belong in a group).

Opportunities for improvement remain. The DRs contain many snippets of difficult text like part numbers and document numbers that pre-processing could remove. Aerospace domain jargon causes problems for the parser. An example of this is the word "safe," which in common usage can act as either a noun or an adjective. However, in the aerospace domain, safe is commonly used as verb, meaning "to make safe." The Charniak parser will consistently fail to tag safe as a verb, thus causing errors for the scoping algorithm, which cannot recover the scope if safe is not identified as a verb.

TABLE I. Precision and Recall, Two Data Sets

Parser	t_p	f_p	f_n	Precision	Recall
Insufficient Clearance Data Set					
Old Parser	18	22	39	0.450	0.316
New Parser	31	11	26	0.738	0.544
2006 Discrepancy Reports Data Set					
Old Parser	64	39	18	0.621	0.780
New Parser	69	19	13	0.784	0.841

Note: Frequencies and frequency totals greater than 60 are due to records with multiple analyst tags.

Further scoping improvements from the new parsing algorithm will produce more information for STAT tagging. Examination of some failed tags showed new types of problems that need to be covered in the ontology. Other ontology entries are tagged too broadly and need further specialization.

The improved STAT parsing can apply to mining other types of text. Promising targets include requirements, hazard analyses and Failure Mode and Effects Analysis worksheets. Extracted failure information and architecture information can be used to construct fault-propagation models. Work is in progress on this extraction and model construction.

Acknowledgment

The authors thank Roger Schwarz, Linda Bromley, Ken Jenks and other members of the NASA Johnson Space Center DR Trend Analysis and Integration Team for their support of the prototyping that has led to these results.

References

[1] A. N. Srivastava and B. Zane-Ulman, "Discovering recurring anomalies in text reports regarding complex space systems," in 2005 IEEE Aerospace Conf. Proc., Big Sky, MT, USA, March 2005.

[2] J. Malin and D. Throop, "Basic concepts and distinctions for an aerospace ontology of functions, entities and problems," in 2007 IEEE Aerospace Conf. Proc., Big Sky, MT, USA, March 2007.

[3] D. Throop, "Reconciler: Matching terse English phrases," in Proc. 2004 Virtual Iron Bird Workshop, NASA Ames Research Center, April, 2004.

[4] J. Grimshaw, Argument Structure. Cambridge, Mass.: MIT Press, 1990.

[5] E. Charniak, "A maximum-entropy-inspired parser," in Proc. NAACL-00, 2000.

[6] F. Gomez, "Building verb predicates: a computational view," in Proc. 42nd Annual Meeting Assoc. Computational Linguistics. Assoc. Computational Linguistics, Morristown, NJ, USA, 2004.