Experiments in Automated Identification of Ambiguous Natural-Language Requirements

Allen P. Nikora
Jet Propulsion Laboratory,
California Institute of Technology
Pasadena, CA
Allen.P.Nikora@jpl.nasa.gov

Jane Huffman Hayes
Computer Science Department,
University of Kentucky
Lexington, KY
hayes@cs.uky.edu

E. Ashlee Holbrook
Computer Science Department,
University of Kentucky
Lexington, KY
ashlee@uky.edu

Abstract — **Recent research indicates that a significant proportion of the failures observed in operational space mission systems can be traced back to ambiguous requirements. Frequently, the functionality and behavior specified by the defective requirement is insufficiently detailed (e.g., information about the expected operational context is missing) or the requirement is phrased so as to permit multiple interpretations. Recent analysis of several thousand anomalies observed in operational space missions shows that about 20% are due to either operator error or faulty procedures. Again, many of these anomalies can be traced back to requirements exhibiting the same types of ambiguities.**

Since a typical space mission is specified by several thousand requirements, a significant amount of effort may need to be expended to discover ambiguous requirements. Many development organizations still conduct this type of discovery manually, after the requirements have been specified – in addition to requiring significant effort, manual analysis is also error-prone.

Previous work has shown the utility of machine learners and simple natural language processing techniques (e.g., parts of speech tagging) for supporting the identification of a relatively small number of temporal requirements within a large body of system requirements. We extend this work to the problem of identifying the ambiguous requirements within the complete set of requirements for the flight and ground components of a recently-launched space mission. Initial results indicate that these techniques can be effective in automatically identifying ambiguous requirements. Unlike heuristic techniques, new learning models for identifying ambiguous requirements can be developed as mission requirements and vocabularies evolve.

Keywords – requirements analysis, requirements ambiguity, text processing.

I. INTRODUCTION

Defective requirements have been shown to be a significant root cause of failures observed in software systems during fielded use [1], [2]. Ambiguous requirement are an important type of defect contributing to operational failures of software systems. A number of software development practices, such as formal inspections [3], are designed to identify this and other types of defects during development so as to prevent their escape into operations. Requirements that have been developed with a specification language (e.g., Promela [4], UML [5]) can be input to tools that will reason about specific properties of the requirements and identify those properties that are not satisfied by the requirements. Although these techniques can be quite effective in identifying some types of requirements defects, they still have a low level of acceptance in software development organizations. A significant amount of effort can be required to learn:

- A formal specification language
- Effective use of the analysis tools.
- The skill of abstracting away unnecessary detail when specifying requirements in a formal language.

At the same time, there is increasing pressure to minimize the cost and schedule time for software development efforts, giving software developers even less time to learn effective use of formal techniques. Consequently, a large majority of software requirements we have seen continue to be specified in natural language.

When analyzing requirements, then, we must be able to deal with natural language requirements. This paper describes work in one specific area, that of identifying ambiguous requirements. Because specification documents for the types of systems with which the investigators have experience typically contain thousands of requirements, manual identification (i.e., reading) of ambiguous requirements may require significant effort, and can also be error-prone. Automated support for identifying ambiguous requirements could reduce the required effort, and potentially increase the identification accuracy. Results obtained to date indicate that well-performing classifiers for identifying ambiguous requirements may be developed using simple representations of the text and structure of natural language requirements.

The remainder of the paper is organized as follows:

- Section II describes work related to identifying ambiguous natural-language requirements.
- Section III describes an approach to identifying ambiguous natural-language requirements.
- Section IV presents the results of our work.
- Section V addresses potential threats to the validity of this work.
- Section VI discusses the results of this work, and lays out directions for future research in this area.

II. Related Work

Hayes et al. have developed several requirements tracing methods and tools for NASA, such as keyword matching based Software Automated Verification and Validation and Analysis System (SAVVAS) Front End Processor (SFEP) [6] and Information Retrieval (IR) based approach [7], [8]. To determine whether a higher-level requirement is related to a lower-level requirement, the IR based requirements tracing method counts the number of common terms and their frequency in both requirements.Then their similary is determined based on the relative frequencies of common terms. This approach has been applied to a small project and has shown improvement over manual tracing. Although this technique can be used to identify related requirements within a subset of requirements of a specific type (e.g., temporal requirements), its intended use is not to identify requirements of a specific type.

The Software Assurance Technology Center (SATC) at the Goddard Spaceflight Center (GSFC) [9] has developed the Automated Requirements Measurement (ARM) tool for assessing the quality of requirements specified in natural language [10]. ARM searches a requirements document for terms that research conducted by the SATC has identified as quality indicators; several of the measured terms are linked by the research to requirements ambiguity.

Requirements Assistant, a commercially available tool, encodes the experience of its developers in a knowledge base geared towards identifying ambiguous, inconsistent, or incomplete requirements [11].

Cobleigh et al. have developed the Property Eludicator (PROPEL), a technique and toolset for elucidating rigorous system properties in natural language [12], [13], [14], [15]. Since the goal of PROPEL is the accurate and complete specification of system properties, it can be considered as a technique and tool for addressing ambiguity of specific types of requirements. Unlike the previously mentioned techniques and tools, PROPEL deals with ambiguity from the point of view of preventing the generation of ambiguous requirements rather than identifying them. The PROPEL website [15] states that "The PROPEL approach provides templates that explicitly capture these [subtle but important, and often unconsidered] details for property patterns that commonly occur in the properties that are created for model checking and other types of analysis. With PROPEL, users are shown the evolving property specification in both "disciplined" English [a restricted subset of natural language] sentences and graphical finite-state automata (FSA), allowing the specifier to easily move between these two views as they develop their properties."

Finally, one of the investigators has recently used machine learning and natural language processing to identify temporal requirements within a specification document [16] and to differentiate between 8 frequently-used Linear Temporal Logic (LTL) patterns in natural-language temporal requirements [17]. Although that work did not address the issue of ambiguous requirements, the goal was to distinguish between two or more different types of natural-language requirements, which is relevant to the work reported here.

III. Data Sources, Representation, and Approach

The requirements we analyzed are for a JPL-managed robotic planetary exploration mission that is currently operating. Nearly 7500 requirements were available for analysis – these included functional, nonfunctional, and development process requirements. For this work, we sampled this set of requirements to produce a set of training data to which machine learners and natural language processing techniques could be applied. The sampling strategy selected over 500 requirements from the original set, each requirement having the same likelihood of being selected as any other. We then read each requirement to classify it as "ambiguous "or "not ambiguous." We are confident in our classifications because the individual who led this subtask (Nikora) has extensive experience (over 30 years) in developing space mission system software and assessing its quality. A small fraction of the requirements could not be classified with confidence; these were excluded from the final set used to construct the training data, which still included over 500 requirements. In the final set of requirements, the number tagged as ambiguous was somewhat larger than, but close to, the number of requirements tagged as ambiguous.

The training data to which we applied machine learners was constructed from the final set of requirements in the following manner:

1. The Trigrams 'n' Tags (TnT) parts-of-speech (PoS) tagger [18], [19] was applied to each word in each requirement as in our earlier work (e.g., [16], [17]). For each text string input to TnT, a string of parts-of-speech tags is returned, each tag corresponding to a word in the original text string. For example, applying TnT to the text "The star scanner shall be calibrated no earlier than 30 days after launch and no later than 40 days after launch" produces the following string of PoS tags: "DT, NN, NN, MD, VB, VBN, RB, RBR, IN, CD, NNS, IN, NN, CC, RB, RBR, IN, CD, NNS, IN, NN". The meanings of the tags in this string are as follows. CC: Coordinating conjunction; CD: Cardinal number; DT: Determiner; IN: Conjunction, subordinating; MD: Modal Verb; NN: Noun, singular or mass; NNS: Noun, plural; RB: Adverb; RBR: Adverb, comparative; VB: Verb, base form; VBN: Past participle. The Penn Treebank

Guidelines [20] define the meanings of all tags produced by TnT.

2. For each word in a requirement, the PoS tag produced by TnT for that word was appended to that word. For the requirement "The star scanner shall be calibrated no earlier than 30 days after launch and no later than 40 days after launch," this operation yields the symbol string "The_DT star_NN scanner_NN shall_MD be_VB calibrated_VBN no_RB earlier_RBR than_IN 30_CD days_NNS after_IN launch_NN and_CC no_RB later_RBR than_IN 40_CD days_NNS after_IN launch_NN."
3. After the PoS tags were appended to the words in the text of the requirement, those symbols containing frequently-used words were removed from the text. A stop list posted on the website of the University of Glascow's Department of Computer Science defines the words that were removed [21]. In addition to the words on this list, symbols containing the word "shall" were removed from the text.
4. Each requirement was formatted as a text string in the Attribute-Relation File Format Attribute-Relation File Format (ARFF) [22], [24] for the WEKA data mining tool [23].
5. WEKA's capabilities were used to transform the text string into a vectorized representation. The text string was converted into three different representations – word/symbol counts, frequencies, and TF (Term Frequency) x IDF (Inverse Document Frequency)
6. We also created representations of the temporal requirements text without the PoS information by omitting steps 1 and 2 above.

Each record in the data section is a vectorized representation of the text string for an individual ISA. In the example above, the first item in the first data record is the value of TFxIDF for the word "parameters", the second item is the value of TF x IDF for the word "schedule", and so forth. For the other representations, these values are word counts and frequencies. The class of the ISA is appended to end of the vector. For the example above, we distinguish between ambiguous requirements (Ambig) and unambiguous requirements (Unambig).

We then developed five variants for each training set by varying the number of included attributes. In this case, an attribute is the count, frequency, or TFxIDF value for the unique symbols or words in the set of requirements. The attributes for each variant were chosen by applying WEKA's implementation of the Information Gain (InfoGain) attribute evaluator in conjunction with the Ranker attribute ranking technique [22]. The first variant of the training data included the attributes accounting for the first 70% of the classification merit according to InfoGain, the second included the attributes accounting for the first 75% of the classification merit, and so on to the fifth variant, which included 90% of the attributes respectively.

Twenty-nine classifiers implemented in WEKA, shown in **TABLE I**, were applied to these training sets. The classifiers were applied to each variant using 10-fold cross validation, the results of which were used to plot Receiver Operational Characteristic (ROC) curves [22] curves for each classifier applied to each variant.

TABLE I. CLASSFIERS APPLIED TO TRAINING SETS

AD Tree	Naïve Bayes Updateable
Bayes Net	Naïve Bayes
Complement Naïve Bayes	NB Tree
Conjunctive Rule	Nnge
Decision Stump	OneR
Decision Table	PART
Hyper Pipes	Random Forest
IB1	Random Tree
IBk	RBF Network
J48	Ridor
JRip	SMO
Kstar	VFI
Simple Logistic	Voted Perceptron
LWL	Zero R
Naïve Bayes Multinomial	

IV. RESULTS

Classifiers are evaluated according to four criteria: pd (probability of detection), pf (probability of false detection, or "false positives"), accuracy, and precision. These criteria are defined with respect to a confusion matrix, as follows:

- Probability of detection (pd): a/(a+b)
- Probability of false detection (pf): c/(c+d)
- Precision: c/(a+c)
- Accuracy: (a+d)/(a+b+c+d)

where a, b, c, and d are entries in a confusion matrix:

Detected as type "x"	**Detected as type "y"**	
a	b	**Really type "x"**
c	d	**Really type "y"**

In this case, we chose as the best classifier the one whose ROC curve comes closest to the performance ideal – a *pd* value of 1 and *pf* value of 0.

Figure 1. - Figure 4. show the ROC curves for the "best" classifiers obtained for each type of representation. Figure 1. and Figure 2. show the performance of the best classifiers for counts, frequencies, and TFxIDF values using text plus PoS tags; Figure 3. and Figure 4. show this same information for representations using text alone. Comparing Figure 1 and Figure 2. with Figure 3. and Figure 4. indicates

that the classifiers using text and PoS information appear to provide better performance in terms of *pd* and *pf* than classifiers using text information alone. However, the difference in performance is not great; further investigation is continuing to determine whether using PoS information in this problem domain will improve classifier performance to the same extent as noted in earlier work (e.g., [17]). Figure 1 and Figure 2. also indicate that the TFxIDF representation performs better than either counts or term frequencies, since the ROC curves for TFxIDF come the closest to the upper left corner of the plot.

TABLE II shows the confusion matrices of the best performing classifier for each of the five variants of the three representations of the training set. For example, the best performing classifier for word counts is SMO, Platt's sequential minimal optimization algorithm for training a support vector classifier [25]. Columns 3 and 4 show the confusion matrices resulting from applying this classifier to the training data for the variants consisting of the attributes accounting for 70%, 75%, 80%, 85%, and 90% of the cumulative classification merit. Columns 5 and 6 contain the classification matrices of the best performing classifier for symbol counts, Complement Naïve Bayes [26]. It is interesting to note in **TABLE II** that there are roughly as many true ambiguous requirements in the training set as there are true unambiguous requirements. If the requirements we analyze for other missions have similar proportions of ambiguous and unambiguous requirements, this suggests that requirements analysts and assurance engineers will benefit from classifiers that perform well in differentiating between ambiguous and unambiguous requirements.

TABLE II also shows that the best performing classifiers are consistent with what we have seen in our earlier work analyzing anomaly reports [16] and temporal requirements [17]. The best performing classifiers in this case are Complement Naïve Bayes in 4 of 6 cases, and SMO in the other two cases. In our work on analyzing anomalies, Complement Naïve Bayes and Multinomial Naïve Bayes were the best performing classifiers in terms of the distance of (pf, pd) from the point (0,1) on the ROC curve. For our earlier work on analyzing temporal requirements, the best performing classifiers in 4 out of 8 cases were SMO and Complement Naïve Bayes.

TABLE III compares the *pd* and *pf* values of the best classifier for each representation. Lines that are bolded indicate the representation that produces the best performing classifier for each type of requirement (ambiguous or unambiguous). For example, the first bolded entry in Table III indicates that the best classifier for ambiguous requirements, having *pd* of 0.807 and *pf* of 0.308, uses both text and PoS tags and is vectorized as TFxIDF. For unambiguous requirements, the classifier developed from the TFxIDF representation of the requirements text using PoS tags performs better than those developed from text only. In this case, the best classifier has a *pd* of 0.692 and *pf* of 0.183.

TABLE II. CONFUSION MATRICES FOR BEST CLASSIFIERS – WITH AND WITHOUT POS INFORMATION

% Cum Classification Merit	Actual Class	Text Only		Text & PoS Tags	
		Detect Ambig.	Detect Unamb	Detect Ambig.	Detect Unamb
Counts					
		Best classifier: SMO		Best classifier: Complement Naïve Bayes	
70	Ambig	225	54	211	63
	Unambig	95	132	95	132
75	Ambig	221	58	201	73
	Unambig	94	133	88	139
80	Ambig	223	56	208	66
	Unambig	82	145	81	146
85	Ambig	219	60	208	66
	Unambig	84	143	74	153
90	Ambig	202	77	201	73
	Unambig	78	149	80	147
Frequencies					
		Best classifier: Complement Naïve Bayes		Best classifier: SMO	
70	Ambig	211	63	219	60
	Unambig	95	132	90	137
75	Ambig	201	73	203	76
	Unambig	88	139	70	157
80	Ambig	208	66	202	77
	Unambig	81	146	71	156
85	Ambig	208	66	206	73
	Unambig	74	153	63	164
90	Ambig	201	73	192	87
	Unambig	80	147	71	156
TFxIDF					
		Best classifier: Complement Naïve Bayes		Best classifier: Complement Naïve Bayes	
70	Ambig	211	63	219	60
	Unambig	95	132	80	147
75	Ambig	201	73	224	55
	Unambig	88	139	78	149
80	Ambig	208	66	228	51
	Unambig	81	146	70	157
85	Ambig	208	66	226	53
	Unambig	74	153	72	155
90	Ambig	201	73	216	63
	Unambig	80	147	72	155

TABLE III. PD AND PF FOR BEST CLASSIFIERS OF REQUIREMENTS – WITH AND WITHOUT POS INFORMATION

Representation	Text Only		Text & PoS Tags	
	pd	*pf*	*pd*	*pf*
Ambiguous Requirements				
Count	0.799	0.361	0.759	0.326
Frequency	0.759	0.326	0.738	0.278
TFxIDF	0.759	0.326	**0.817**	**0.308**
Unambiguous Requirements				
Count	0.639	0.201	0.674	0.241
Frequency	0.674	0.241	0.722	0.262
TFxIDF	0.674	0.241	**0.692**	**0.183**

TABLE IV. NUMBER OF ATTRIBUTES IN TRAINING SET VARIANTS

	Percentage of Cumulative Ranking Merit				
Representation for which Training Set is Constructed	**70%**	**75%**	**80%**	**85%**	**90%**
Word Only Representations 2704 attributes in unmodified training set					
Counts	427	506	651	706	887
Frequency	382	482	546	693	778
TFxIDF	382	482	546	693	778
Words and PoS Tags Representations 2957 attributes in unmodified training set					
Counts	455	566	665	811	911
Frequency	476	550	627	755	897
TFxIDF	476	550	627	755	897

TABLE III compares the *pd* and *pf* values of the best classifier for each representation. Lines that are bolded indicate the representation that produces the best performing classifier for each type of requirement (ambiguous or unambiguous). For example, the first bolded entry in Table III indicates that the best classifier for ambiguous requirements, having *pd* of 0.807 and *pf* of 0.308, uses both text and PoS tags and is vectorized as TFxIDF. For unambiguous requirements, the classifier developed from the TFxIDF representation of the requirements text using PoS tags performs better than those developed from text only. In this case, the best classifier has a *pd* of 0.692 and *pf* of 0.183.

We see that for both ambiguous and unambiguous requirements, the representation for which the ROC curve comes closest to the point of ideal classifier performance is TFxIDF. We also see that representations that include PoS tags produce classifiers that perform better than those that do not. These results are consistent with our earlier work showing that using PoS information and TFxIDF representations seem to yield better performing classifiers than counts and frequencies.

TABLE IV specifies the number of attributes in each of the five variants for each representation of the training set. For example, the count-based representation using both text and PoS tags containing those attributes accounting for 75% of the cumulative classification merit contains 665 attributes. If no attributes had been removed, there would be 2957 attributes in the training set. The best classifier performance was found using the text and PoS tag training set containing the attributes accounting for the first 85% of cumulative ranking merit, meaning that approximately ¼ of the attributes from the original training set were required to build a well-performing classifier.

TABLE V. NUMBER OF ATTRIBUTES IN TRAINING SET VARIANTS IN PREVIOUS WORK IN ANALYZING ANOMALIES

	Percentage of Cumulative Ranking Merit					
Anomaly Report Class	**70%**	**75%**	**80%**	**85%**	**90%**	**100%**
Word Counts Representations						
Flight software (FSW)	102	309	341	361	372	386
Ground software (GSW)	123	139	157	362	375	386
FSW+GSW	342	355	366	372	378	386
Procedural or Operator error (PROC)	81	94	114	145	211	386
TFxIDF Representations						
FSW	48	54	61	69	83	378
GSW	83	95	107	120	134	392
FSW+GSW	78	89	102	116	132	385
PROC	55	61	75	111	148	416

It is interesting to compare this with our earlier work in analyzing temporal requirements and anomaly reports. For example, **TABLE V** specifies the number of attributes in the various training sets we constructed in analyzing anomaly reports to more accurately identify software-involved anomalies [16]. The number of attributes in the unmodified training sets is given in the last column of **TABLE V**; for the word counts representation (no PoS tags), there are a total of 386 attributes; for the text only TFxIDF representation, the number of attributes in the unmodified training set varies from 378 to 416 – this is because of the way in which we applied the WEKA-implemented classifiers to the training data. For this work, the TFxIDF representations also produced the best classifier; since the best classifier performance was seen with training data containing only those attributes accounting for the first 80% or 85% of the cumulative ranking merit, between 1/6 and ¼ of the total number of attributes in the unmodified training data set. In our related work on analyzing temporal requirements [17], there were somewhat over 900 attributes in the unmodified training set. For the TFxIDF text and PoS representation, the best performance was often obtained for those training sets containing 500-700 attributes, over half of the number

of attributes in the unmodified training set. The number of attributes in the unmodified training set appears to distinguish this particular problem of identifying ambiguous requirements from our earlier work.

V. Potential Threats to Validity

One potential threat to the validity of the work reported here concerns the construction of the training sets used to develop learning models for distinguishing ambiguous and unambiguous requirements. As mentioned in Section III, the training sets were constructed by sampling the original set of ~7500 requirements to obtain a subset of ~500 requirements from which the training data would be constructed. To obtain the subset, the original requirements were first listed in the order in which they were extracted from the requirements repository. Every 16th requirement was then retained for potential inclusion in the training set. Our intent in constructing the training set in this manner was to include requirements for all aspects of the mission being analyzed. It is possible, although unlikely, that the subset of requirements extracted in this manner does not represent the true proportion of ambiguous and unambiguous requirements in the original set of requirements extracted from the repository. As part of our ongoing work, however, we will perform additional samplings of the requirements to determine whether the proportions of ambiguous and unambiguous requirements in training sets are representative of the complete set of mission requirements.

Another potential threat concerns the classification of the requirements used to construct the training set. The classification, led by one of the co-authors (Nikora), was performed manually by reading each requirement to determine whether it was ambiguous, unambiguous, or unclassifiable. Those requirements that were deemed by the manual reading to have either more than a single meaning or a meaning that could not be determined (e.g., insufficient information), were labeled as "ambiguous" in the training data. Those requirements for which the readers could only agree on a single interpretation were labeled as "unambiguous" in the training data. There were requirements for which a classification could not be made; these were not included in the training data. The readers may have made errors of interpretation – for example, they may have labeled an ambiguous requirement as "unambiguous" because they were not aware of a second meaning for that requirement. They may also have labeled unambiguous requirements as "ambiguous" because they did not have the necessary domain knowledge to correctly interpret the requirement in the context for which it was intended. In cases for which the classification lead recognized such a lack of sufficient domain information, contact was made with technical staff in the appropriate domains to help determine the classification of the requirement in question.

VI. Discussion

Our results to date indicate that relatively simple machine learning and natural language processing techniques may be useful in providing automated support to mission developers and assurance engineers for identifying ambiguous requirements within a set of specification documents. Since a large proportion of the requirements we have analyzed so far appears to be ambiguous, and previous analyses of anomaly reports [2] indicates that a significant proportion of software-related anomalies are related to misunderstood or missing requirements, this type of automated support may help reduce the number of requirements defects propagated into the implemented system. Relatively simple machine learning and natural language processing techniques produce classifiers providing a detection rate high enough that they might be used in real development efforts, although the number of false positives may still make their use somewhat impractical. However, if the requirements we are analyzing for other missions have similar proportions of ambiguous and unambiguous requirements as those we have analyzed for the work reported here, only a modest improvement in performance may be required to develop classifiers that can support requirements analysis for real projects. Future work will investigate more sophisticated learning techniques (e.g., voting, bagging, boosting) and additional data representations (e.g., inclusion of additional syntactic information) to reduce the false positive rate. We will also investigate how other natural language processing techniques in addition to parts-of-speech tagging may be used to help identify ambiguous requirements more accurately. For example, the number of ways a requirement may be parsed with a natural language parser may be an indicator of a particular type of ambiguity. To address the potential threat of incorrectly classifying requirements as ambiguous or unambiguous, the investigators will also use existing guidelines for producing unambiguous requirements (e.g., [28]) to assure the correctness of manual classifications.

We saw that the number of attributes in the unmodified training set was substantially larger than the number of attributes in the training sets developed for our earlier work in analyzing anomaly reports and temporal requirements. As we add additional requirements from the mission we have been analyzing as well as from additional missions, the number of attributes may increase substantially as the vocabulary of the requirements grows. If the number of attributes becomes too large, the computational resources required to construct well-performing learning models could reduce the utility of this technique. To minimize the possibility of this situation arising, we will investigate techniques to reduce the number of attributes required to construct well-performing learning models (e.g., discretization, additional attribute evaluation methods, etc.).

As we examine additional requirements in future work, there may be variations in the requirements across missions

due to differences in the development teams producing the requirements. For this investigation, a single team was responsible for the requirements that were analyzed. However, as we acquire requirements from multiple missions and analyze them, it will be necessary to determine whether the composition of different missions' development teams has an effect on the learning models produced.

This work represents an early stage of a multi-year effort to investigate ways of automating the identification of ambiguous and inconsistent natural-language requirements. We will investigate the identification of ambiguous requirements in more detail. For example, we will investigate techniques for identifying different types of ambiguity – lexical, syntactic, and semantic. Lexical ambiguity occurs when there is insufficient context within the text to narrow its scope to one meaning. Syntactic ambiguity is found in text that may be parsed in multiple ways, leading to multiple meanings. Semantic ambiguity, often referred to as vagueness, is found in sentences with concepts that may have multiple meanings based on the formality, surrounding text, or context of a situation. We will also analyze trends in the proportions of the types of ambiguous requirements developed for different missions, as was recently done for different types of defects identified in space mission on-board software during flight operations [27].

ACKNOWLEDGEMENT

The work described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology. This work was sponsored by the National Aeronautics and Space Administration's Office of Safety and Mission Assurance (OSMA) Software Assurance Research Program (SARP).

REFERENCES

[1] R. Chillarege, W. L. Kao, and R. G. Condit, "Defect Type and its Impact on the Growth Curve," in the 13th IEEE International Conference on Software Engineering, Austin, Texas, May 13-17, 1991.

[2] R. Lutz and C. Mikulski, "Orthagonal Defect Classification for Projects," in the JPL/GSFC Quality Mission Software Workshop, Rehoboth Beach, DE, May 13, 2003.

[3] F. Shull and C. Seaman, "Inspecting the History of Inspections: An Example of Evidence-Based Technology Diffusion," IEEE Software. 25, 1 (Jan. 2008), 88-90.

[4] Gerard Holzmann, The SPIN Model Checker Primer and Reference Manual, Addison-Wesley, 2003, ISBN 0-321-22862-6.

[5] OMG Unified Modeling Language™ (OMG UML), Superstructure version 2.2; Object Management Group, Feb. 2009.

[6] J. H. Hayes, "Risk Reduction Through Requirements Tracing," Proc. of 1990 Software Quality Week, San Francisco, CA, 1990.

[7] J. H. Hayes, A. Dekhtyar, J. Osbourne, "Improving Requirements Tracing via Information Retrieval," in Proceedings of the 2003 IEEE International Conference on Requirements Engineering, Monterey, California, Sept. 2003.

[8] J. H. Hayes, A. Dekhtyar, S. Sundaram, S. Howard, "Helping Analysts Trace Requirements: An Objective Look," in Proceedings of the IEEE Requirements Engineering Conference (RE) 2004, Kyoto, Japan, Sept. 2004, pp. 249-261.

[9] Software Assurance Technology Center (SATC), Goddard Space Flight Center, National Aeronautics and Space Administration, http://satc.gsfc.nasa.gov/index.php, last viewed Feb. 19, 2010.

[10] W. Wilson., L. Rosenberg, and L. Hyatt, "Automated Quality Analysis of Natural Language Requirement Specifications," in the Proceedingsof the 14th Annual Pacific Northwest Software Quality Conference, Portland, 1996.

[11] Requirements Assistant, http://www.requirementsassistant.nl, last viewed may 23, 2010.

[12] R. L. Cobleigh, G. S. Avrunin, L. A. Clarke, "User Guidance for Creating Precise and Accessible Property Specifications," in the Proceedings of the ACM SIGSOFT 14th International Symposium on Foundations of Software Engineering (FSE14), Portland, OR, pp. 208-218, Nov. 2006.

[13] R. L. Smith, G. S. Avrunin, L. A. Clarke, "From Natural Language Requirements to Rigorous Property Specifications," Monterey Workshop 2003 (Workshop on Software Engineering for Embedded Systems (SEES 2003) From Requirements to Implementation, Chicago, IL, pp. 40-46, Sept. 2003.

[14] R. L. Smith, G. S. Avrunin, L. A. Clarke, L. J. Oster-weil, "PROPEL: An Approach Supporting Property Elucidation," in the Proceedings of the 24th International Conference on Software Engineering (ICSE 2002), Orlando, FL, pp. 11-21, May 2002.

[15] University of Massachusetts, Laboratory for Advanced Software Engineering Research, PROPEL web page, http://laser.cs.umass.edu/tools/propel.shtml. Last viewed May 23, 2010.

[16] A. Nikora, G. Balcom, "Improving the Accuracy of Space Mission Software Anomaly Frequency Estimates,", proceedings of Third International Conference on Space Mission Challenges for Information Technology (SMC-IT 2009), Jul 2009, Pasadena, CA

[17] A. Nikora, Galen Balcom, "Automated Identification of LTL Patterns in Natural Language Requirements," in the Proceedings of the 20th International Symposium on Software Reliability Engineering, IEEE Press, Nov. 2009 pp. 185-194, doi: 10.1109/ISSRE.2009.15.

[18] T. Brants, "TnT - A Statistical Part-of-Speech Tagger," in the Proceedings of the 6th Applied Natural Language Processing Conference, Apr. 2000, pp. 224-231, doi:10.3115/974147.974178.

[19] T. Brants, "TnT -- Statistical Part-of-Speech Tagging," Universität des Saarlandes, Department of Computational Linguistics, TnT software download application form, http://www.coli.uni-sb.de/~thorsten/tnt/, viewed May 23, 2010.

[20] "Part of Speech Tagging Guidelines for the Penn Treebank Project," The Penn Treebank Project, University of Pennsylvania, http://www.cis.upenn.edu/~treebank/, viewed May 23, 2010.

[21] Stop list, IR linguistic utilities, Idomeneus technology transfer server, University of Glascow Department of Computer Science, http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words, last viewed Feb. 19, 2010.

[22] Ian H. Witten, Eibe Frank, Data Mining: Practical Machine Learning Tools and Techniques, Second Edition, Morgan Kaufmann, June 2005, ISBN 0-12088-407-0.

[23] The University of Waikato Computer Science Department Machine Learning Group, WEKA software download, http://www.cs.waikato.ac.nz/~ml/weka/index.html, last viewed May 23, 2010.

[24] The University of Waikato Computer Science Department Machine Learning Group, Attribute-Relation File Format, http://www.cs.waikato.ac.nz/~ml/weka/arff.html, last viewed May 23, 2010.

[25] J. Platt: Fast Training of Support Vector Machines using Sequential Minimal Optimization. In B. Schoelkopf and C. Burges and A. Smola, editors, Advances in Kernel Methods - Support Vector Learning, 1998

[26] J. D. Rennie, L. Shih, J. Teevan, D. R. Karger: Tackling the Poor Assumptions of Naive Bayes Text Classifiers. In: ICML, 616-623, 2003.

[27] A. Nikora, M. Grottke, K. S. Trivedi, "An Empirical Investigation of Fault Types in Space Mission System Software", to appear in proceedings of 40th IEEE/IFIP International Conference on Dependable Systems and Networks, Chicago, IL, June, 2010.

[28] D. Berry, E. Kamsties, M. Krieger, "From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity," Nov. 2003, available at http://se.uwaterloo.ca/~dberry/handbook/ambiguityHandbook.pdf, last viewed May 23, 2010.

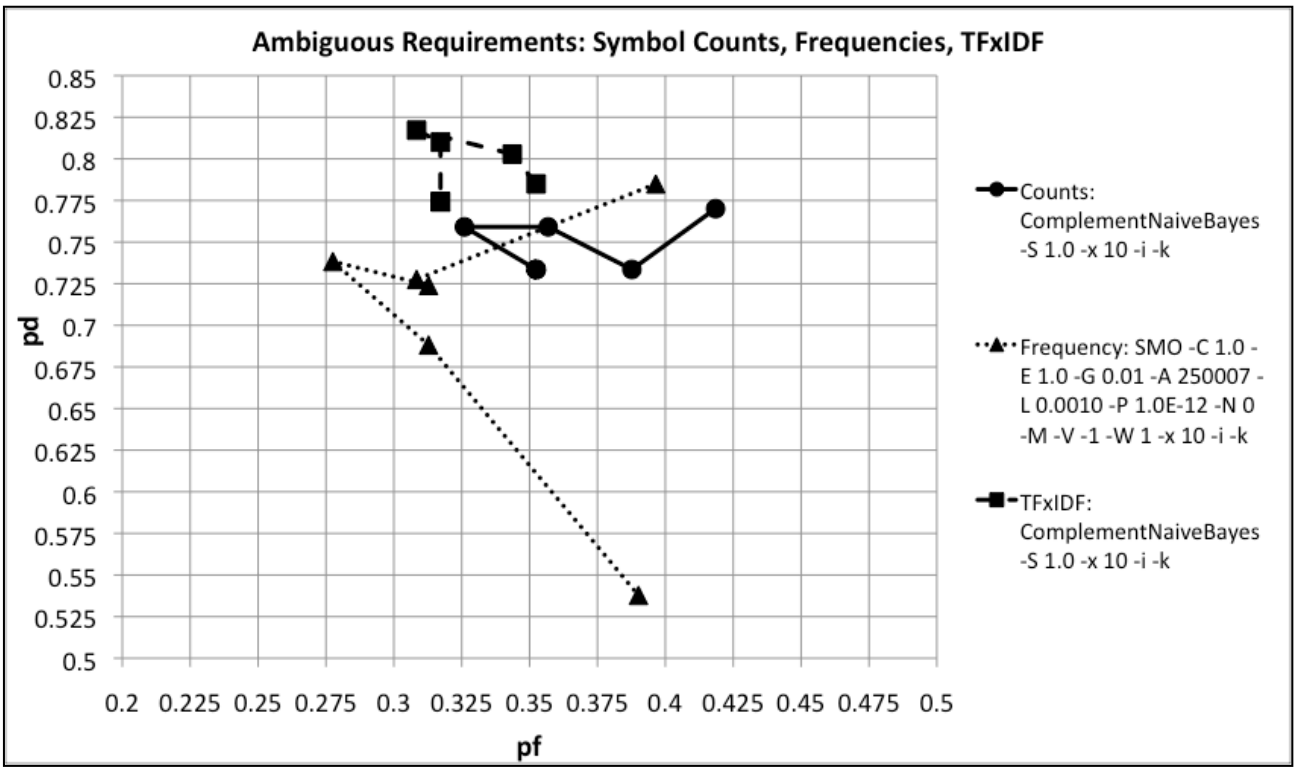

Figure 1. ROC Curves for Ambiguous Symbol Counts, Frequencies, and TFxIDF.

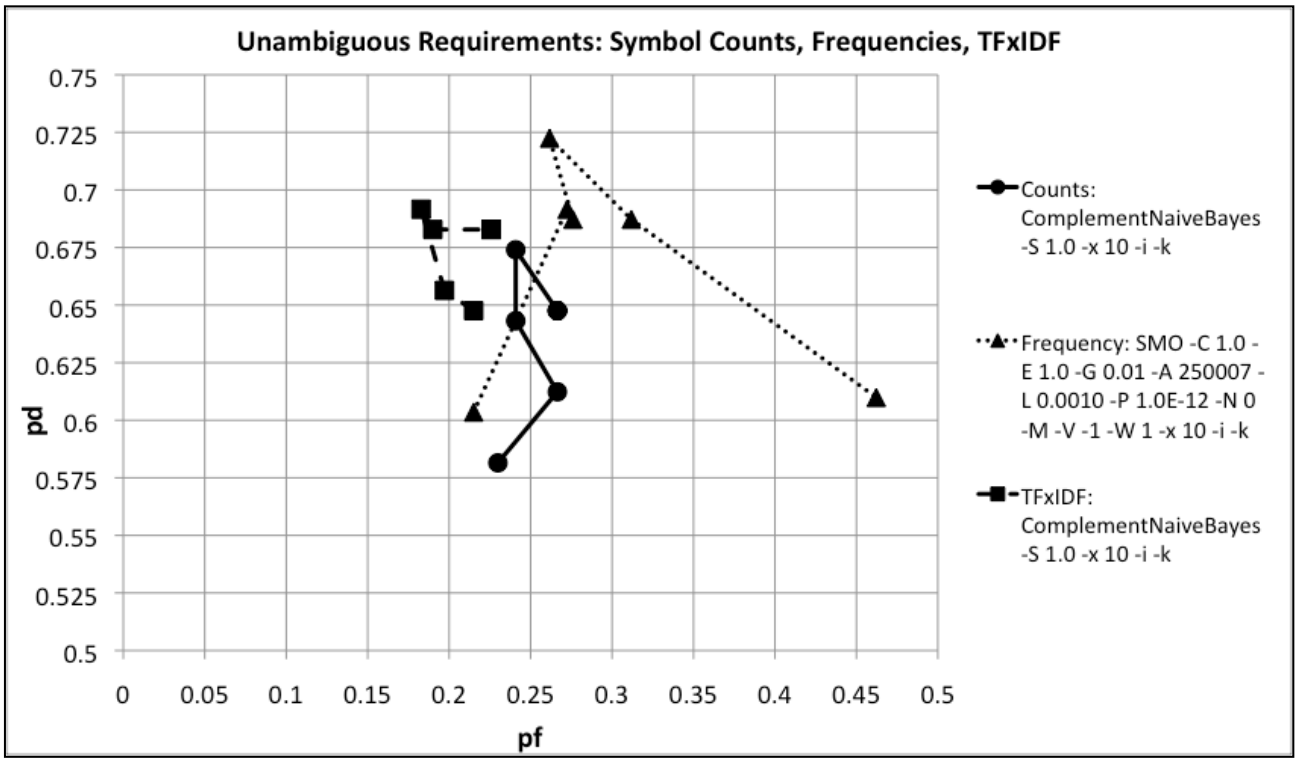

Figure 2. ROC Curves for Unambiguous Symbol Counts, Frequencies, and TFxIDF

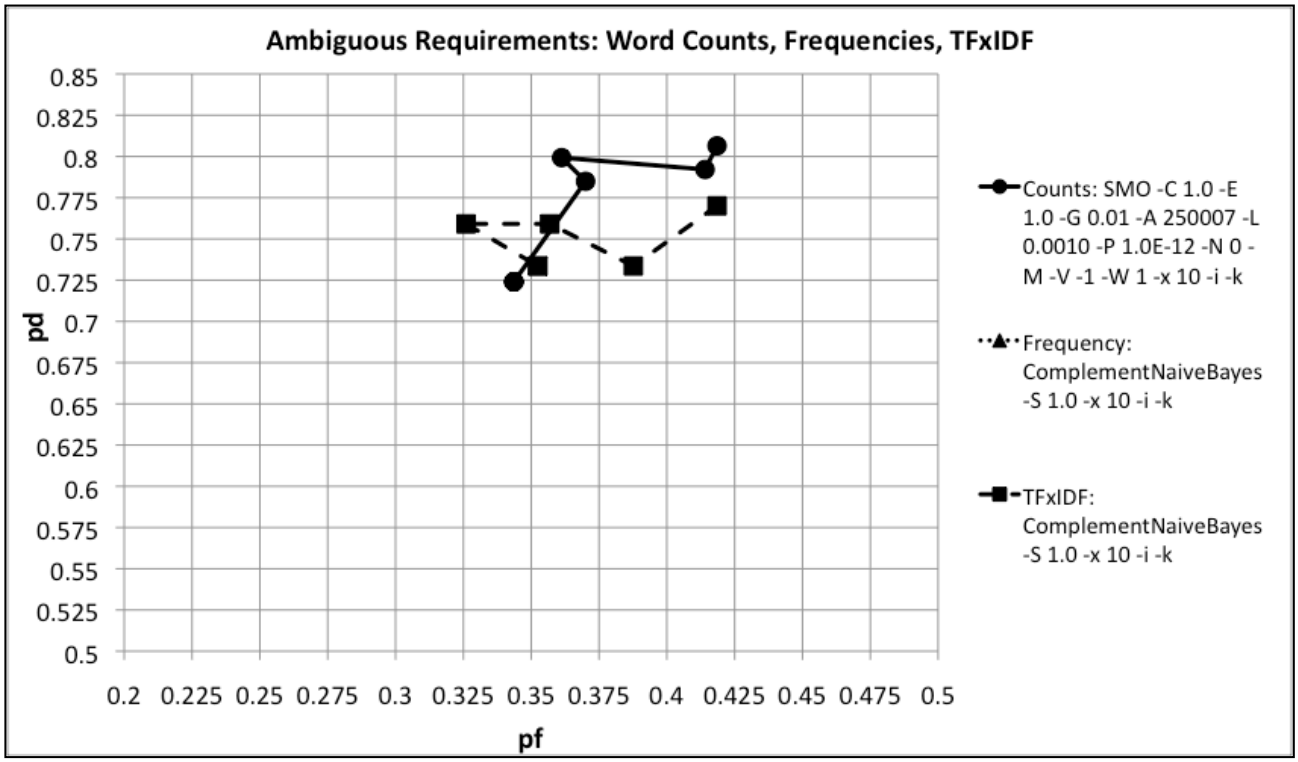

Figure 3. ROC Curves for Ambiguous Word Counts, Frequencies, and TFxIDF

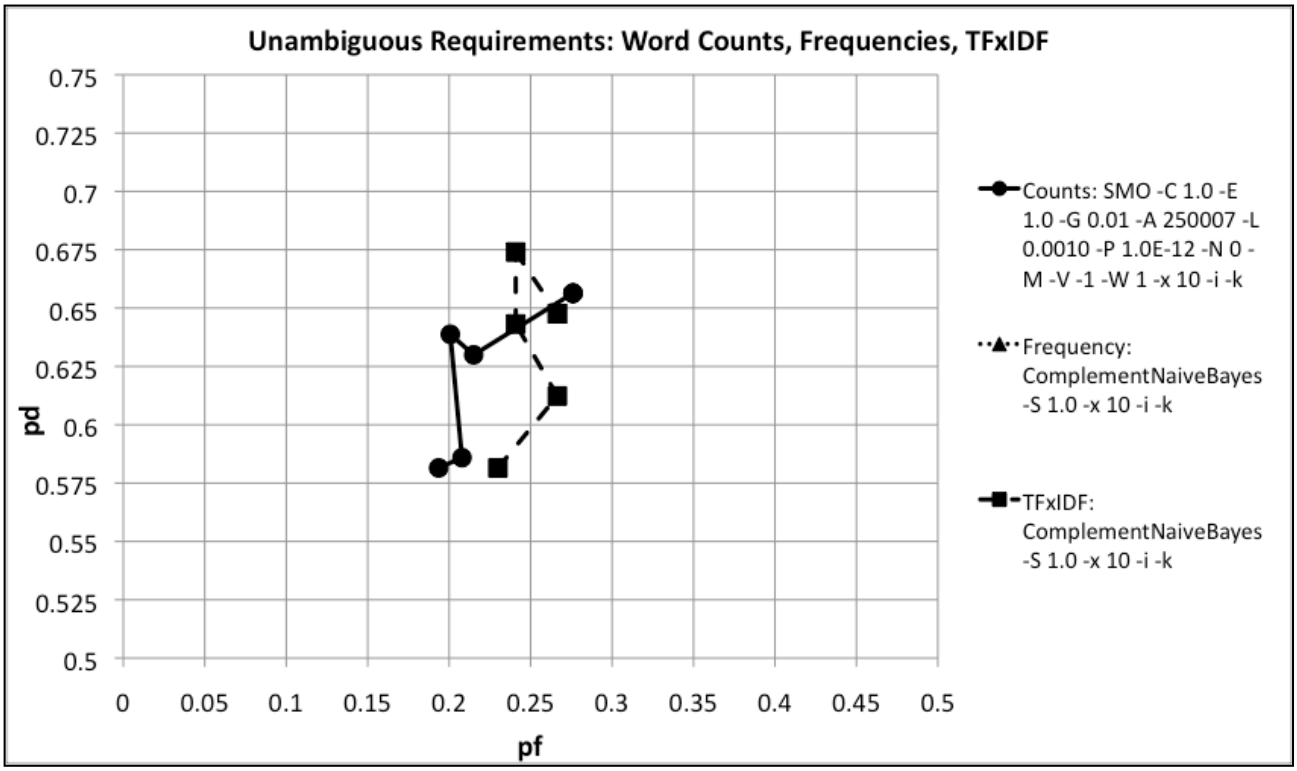

Figure 4. ROC Curves for Unambiguous Word Counts, Frequencies, and TFxIDF