Predicting Verification Methods from Natural Language Requirements

Michael D. Prendergast
Colorado Springs, USA
mike.prendergast@ieee.org

Abstract – **A Verification Cross-Reference Matrix (VCRM) is a table that depicts the verification methods for requirements in a specification. Usually requirement labels are rows, available test methods are columns, and an "X" in a cell indicates usage of a verification method for that requirement. Verification methods include Demonstration, Inspection, Analysis and Test, and sometimes Certification, Similarity and/or Analogy. VCRMs enable acquirers and stakeholders to quickly understand how a product's requirements will be tested.**

Maintaining consistency of very large VCRMs can be challenging, and inconsistent verification methods can result in a large set of uncoordinated "spaghetti tests". Natural language processing algorithms that can identify similarities between requirements offer promise in addressing this challenge.

This paper applies and compares compares four natural language processing algorithms to the problem of automatically populating VCRMs from natural language requirements: Naïve Bayesian inference, (b) Nearest Neighbor by weighted Dice similarity, (c) Nearest Neighbor with Latent Semantic Analysis similarity, and (d) an ensemble method combining the first three approaches. The VCRMs used for this study are for slot machine technical requirements derived from gaming regulations from the countries of Australia and New Zealand, the province of Nova Scotia (Canada), the state of Michigan (United States) and recommendations from the International Association of Gaming Regulators (IAGR).

Keywords—requirements analysis, natural language processing, verification matrices, verification methods, slot machines.

I. INTRODUCTION

Stakeholder requirements often form the basis of product acceptance testing, and a verification cross reference matrix (VCRM) table can be used to document plans and agreements on how each requirement should be tested. VCRMs are tables with requirements labels as rows and verification methods (such as (Demonstration, Inspection, Analysis, and Test) as columns. The test method for a particular requirement is designated by placing an "X" in the cell of the corresponding row and column. The VCRM links requirements engineering and test planning.

Verification methods include (summarizing from [1]):

- Inspection – Visual or dimensional examination of an element, relying on human senses or simple measurements,
- Analysis – Use of analytical evidence obtained from mathematical or logical reasoning and/or simulations without any intervention on the submitted element,
- Demonstration – Show of correct operation of the examined element against operational and observable characteristics without physical measurement, and
- Test – Quantitative verification of measurable characteristics, operability, supportability or performance capability when subjected to controlled conditions.

Other methods that are sometimes included are Analogy or Similarity (forms of analysis comparing one product to a similar product) and Certification (verification by a third party or previous event).

Requirements can often be verified by any of several means. For example, correctly computed quantities can often be verified by either Demonstration on a display, or Test with instrumentation to show the inputs and outputs of a calculation.

On very large programs, a VCRM may include tens of thousands of entries. Verification method consistency is important, because an internally inconsistent VCRM can result in a large collection of uncoordinated "spaghetti tests". Also, for programs that intend to leverage product reuse, an inconsistent VCRM can impair test program reuse efforts.

Natural language processing (NLP) algorithms offer promise in addressing the challenge of maintaining VCRM consistency by grouping requirements that are by some measure similar to each other. This study compares four different NLP approaches for automatically populating VCRMs from natural language requirements.

Section 2 describes prior work in this area and the significance of this contribution.

Section 3 discusses the data sets and their preparation. Section 3.1 describes the data used in this study. Technical requirements were extracted from four sets of slot machine gaming regulations, one from Australia and New Zealand [2], one from Nova Scotia (Canada) [3], one from the International Association of Gaming Regulators [4] and one from Michigan (United States) [5]. Section 3.2 describes the data pre-processing performed on this data. Pre-processing includes requirements transformation and extraction from the regulations, removing stop words and punctuation, and

stemming the remaining keywords. It also includes assigning "truth data" verification methods (VMs) using a rules model and domain knowledge. The result is a "truth VCRM" matrix used to assess model accuracy.

Section 4 describes the models analyzed. Four models were compared: Naïve Bayes, Nearest Neighbor (NN) by weighted Dice similarity, NN with Latent Semantic Analysis (LSA) weights, and an ensemble model.

Analysis of model results is found in Section 5. The three standalone models showed VM accuracy of 77%-83%, and NN with LSA weights performed the best. Better yet was the ensemble model, which had a VM accuracy of 85%. Modeling and human error sources that reduced model accuracy are also analyzed in this section.

Conclusions and next steps are in Section 6. Given the accuracy of these models, they can be used as analysis tools for VCRM consistency checking. None of the four algorithms evaluated were robust enough to replace the engineer with domain expertise, however.

II. Prior Work and This Contribution

Many papers have applied Natural Language Processing (NLP) to requirements engineering, and [6] lists over 400 of them. A high-level summary of recent thrusts to apply NLP to problems in requirements engineering can also be found in [7]. One of these thrusts is similarity analysis.

Similarity analyses within and between engineering documents are found in [8, 9, 10]. Cluster analysis to identify requirement similarities across a product line is found in [11]. Similarity analysis to identify functional and non-functional requirements from user app reviews is found in [12, 13]. A similar effort using requirements derived from mathematical software is found in [14]. NLP was used to identify non-functional requirements from natural language documents in [15]. A similar approach for slot machine development, the technical domain of this paper, is found in [16].

On the test side, [17, 18, 19, 20] have all proposed NLP techniques for auto-generation of test cases. These approaches have had varying degrees of success, as the test problem has the potential for exponential explosion in test case counts. An effort to optimize test cases by clustering is described in [21].

To date, however, similarity analysis has not been used to automatically generate a VCRM. That is the focus of this paper.

III. Data Sets

3.1. Casino Regulation Source Data

Slot machine technical requirements were selected for this study because of the plethora of freely available technical requirements for their construction from regulatory bodies. This study used natural language technical requirements derived from compliance regulations from Australia and New Zealand [2], Nova Scotia [3], the International Association of Gaming Regulators (IAGR) [4], and the state of Michigan [5]. These regulations are described below.

3.1.1. Australia/New Zealand

The *Australia/New Zealand Gaming Machine National Standard 2016* [2] is the national standard of both Australian and New Zealand for electronic gaming machines. It contains both exposition and definition information as well as numerous technical requirements for electronic gaming machines. This includes requirements for payouts, progressive awards, displays ("artwork"), cybersecurity and auditability. The standard also contains process instructions for certifying new electronic gaming machines. This document is one of the most comprehensive gaming machine regulations available, 82 pages long and containing 1,191 paragraphs and sub-paragraphs.

3.1.2. Nova Scotia

Nova Scotia's *Casino Regulations made under Section 127 of the Gaming Control Act* [3] contains both regulations for casino operations in Nova Scotia as well as detailed technical requirements for slot machines. The requirements include requirements for hardware, software, error handling (including tilt conditions), and odds calculations and progressive payouts. The document contains 201 paragraphs and subparagraphs.

3.1.3. International Association of Gaming Regulators

In an effort to consolidate electronic gaming requirements into an international standard, the International Association of Gaming Regulators (IAGR) developed the *International Technical Standards for Electronic Gaming Machines* [4]. This 42-page, 353 paragraph proposed standard includes requirements for hardware, interference effects, software, memory, electronic ledgers ("meters"), communications and artwork. Though it contains a few definitions and some background information, this document is almost entirely comprised of technical requirements.

3.1.4. Michigan

Michigan Gaming Control Board Casino Gaming [5], like the Nova Scotia regulations, serves as both a set of regulations for casino operations as well as a set of detailed technical requirements for slot machines. Almost all of the slot machine technical requirements are found in *Part 8, Conduct of Gaming/Gaming Equipment*, which contains 294 paragraphs and sub-paragraphs. Like the other standards, these requirements include security, error handling, software, hardware, communications and artwork.

3.2. Data Preprocessing

Data preprocessing consisted of three activities. First, the technical requirements had to be extracted from the regulations. Then, VMs were tagged to each requirement. These "truth data" VMs formed the basis by which the accuracy of the four NLP models would be judged. Finally, the requirement text was prepared for analysis by removing stop words and punctuation, stemming the remaining words, and discarding one- and two-letter words.

3.2.1. Requirements Extraction

Gaming regulations are written in natural language by administrators, lawyers or regulators, and often lack the word "shall". Requirements extraction from the regulations was performed in two passes. First, a rule-based heuristic was used to extract candidate requirements. Then, recommendations from

the rule-based heuristic were manually reviewed and updated by a systems engineering requirements expert.

A 12-rule heuristic algorithm that had over 99% accuracy in extracting technical slot machine product requirements from Nevada and South Dakota gaming regulations is described in [16]. That ruleset was adopted for this study but extended to 14 rules so as to take into account phrasing variations across the different English-speaking countries (Canada, New Zealand, Australia and United States). The 14-rule heuristic employed to transform the text into requirement "shall" statements and conditional "may" statements for this study was:

1) "must", "should", "may" in "may not" or "may only", "will"→ "shall",
2) "[is, are] required to", "[is, are] to" → "shall",
3) "can" → "may",
4) "[is, are] (not) permitted", "[is, are] (not) allowed" → "shall (not) be allowed",
5) "cannot" → "shall not"
6) "[is, are] required" → "shall be required",
7) "[is, are] only to" → "shall only"
8) "[is, are] (not) to be" → "shall (not) be"
9) "requires a" → "shall require a",
10) "[is, are] [responsible, determined]"→"shall be [responsible, determined]",
11) "if … are" → "if … shall be",
12) "responsibility … lies" → "responsibility … shall lie",
13) "no … may" → "no … shall"
14) Each bullet in a requirement is itself a requirement.

To illustrate this last rule, consider the requirement:

Each machine shall contain: a) a hopper, and b) a payout meter.

This requirement is transformed into the following two requirements:

Each machine shall contain a coin hopper.

Each machine shall contain a payout meter.

These rules were able to extract operational and technical requirements from the regulations with 98% accuracy. Rule failures occurred when the ruleset tried to create requirements out of definitions and for certain edge-case negative requirements. This simple 14-rule heuristic cannot handle complex requirement rewording, and hence the human-in-the-loop was critical to this process. Consider for example the almost nonsensical regulation from Michigan:

"Direct collection of the random number data from the actual submitted (game) is required unless it is not possible."

On first blush, this requirement may seem to have no meaning at all, equivalent to "do X unless you cannot do X". Though poorly worded, the intent of this regulation is to levy a conditional requirement, with the condition that the requirement applies if the vendor selects a random number generator that can be monitored without interfering with the random number generation process itself. For example, a random number generator relying on quantum processes might not be able to be monitored without affecting its state. A rewording of this requirement to make it clearer might read:

"If the slot machine random number generator permits direct collection of random data without affecting the generator's state, then that data shall be collected, stored and made available for post-test analysis."

This type of complex transformation was beyond the scope of our simple ruleset, and so required a human-in-the-loop.

3.2.3. Setting Verification Method Truth Data

VM selection is somewhat subjective, because a requirement may be verifiable by any of several means. Test engineers may have different preferences for how to test certain types of requirements. Therefore, it was important to have a consistent and repeatable approach for setting VM truth data. The approach selected for this study was:

- Any requirement could be verified by physical inspection was tagged as "Inspection". This includes requirements verified by inspection of the artwork on the game machine.
- For any other requirement, if it is most easily verified through analysis then the VM is set to "Analysis". This includes requirements verified by calculations showing the game machine's payout profile. It also includes statistical analyses, such as calculating the independence of random numbers.
- For any other requirement, if it could be by an actor performing a use case on the game machine without instrumenting the machine, then the VM was set to "Demonstration". This included both requirements involving game play as well as requirements for machine maintenance.
- Requirements that needed instrumentation and detailed examination of captured data to verify them had VMs set to "Test". Examples included requirements for memory and software integrity checks.
- Summary requirements rollups (i.e., those requirements that are completely verified when other, subordinate requirements are verified) and non-technical process requirements for casino operations were tagged as "Not Applicable".

When a requirement needed multiple VMs for full verification, the predominant VM was chosen. When no one method was clearly predominant, the requirement was split into constituent parts, each part having its own VM.

Table 1 shows the distribution of the game machine technical requirements by source and VM at the end of this activity.

Table 1: Verification Method Counts per Regulation Set

Regulation Set	Technical Reqs	Demo	Inspection	Analysis	Test
Australia/New Zealand	618	249	227	88	54
Nova Scotia	75	51	13	8	3
IAGR	222	119	23	47	33
Michigan	177	98	22	18	39

3.3 Requirement Text Preparation

After requirements were extracted and truth VMs specified, the requirement text was prepared for analysis by removing stop

words and punctuation and stemming the remaining words. Only stemmed words at least three letters long were retained.

4. VM Prediction of Michigan Requirements

Each of the four models was trained on Australia/New Zealand, Nova Scotia and IAGR VM data and then the models were used to to predict Michigan VM data. The four models analyzed were: Naïve Bayesian inference, NN by weighted Dice similarity, NN with LSA weights, and an ensemble majority-vote model that combined the results of the other three model. These models are described in the sections below.

4.1. Naïve Bayes Model Description

A Naïve Bayes model used Bayesian inference to estimate the probability that a Michigan requirement belonged to each of the four VM categories: Demo, Inspection, Analysis or Test. The predicted VM was then set to the category with the highest probability.

To illustrate how this model works, let X be a word in the requirements control set, and define the frequency of the wofd X in the control set Demonstration, Inspection, Analysis and Test requirements to be $f_D(X)$, $f_I(X)$, $f_A(X)$, and $f_T(X)$, respectively. Set $f_{tot}(X)$ to be the overall frequency of X in the control set corpus. From Bayes' theorem, for any randomly selected word X and requirement R, the probability that R has a verification method of Demonstration given that it contains word X is:

$$P(R\ is\ Demo|X\ in\ R) = \frac{P(R\ is\ Demo \cap X\ in\ R)}{P(X\ in\ R)}$$

$$= P(X\ in\ R|R\ is\ Demo) * \frac{P(R\ is\ Demo)}{P(X\ in\ R)}$$

$$= f_D(X) * \frac{P(R\ is\ Demo)}{f_{tot}(X)} \quad (1)$$

But all three of the terms in this last expression can be calculated: $f_D(X)$ and $f_{tot}(X)$ are the word frequencies described earlier, and $P(R\ is\ Demo)$ is the fraction of control requirements that have a verification method of demonstration.

Now Suppose that a new requirement not in the control dataset consists of n words, with $m \leq n$ of the words in control requirements. If the m words are denoted by $X_1, X_2, \ldots, X_m$, then the equation

$$P(R\ is\ Demo|X_1, X_2, \ldots, X_m\ in\ R)$$

$$\approx P(R\ is\ Demo) \prod_{i=1}^{m} \frac{f_D(X_i)}{f_{tot}(X_i)} \quad (2)$$

provides an estimate of the probability that R has a VM of Demonstration. (Similar equations can be written for the other VMs.) The $n - m$ words that are in requirement R but not in the control requirement set are ignored in the computation.

Equation (2) is approximate because equality is valid only if the words within the requirement are independent, and this assumption is usually violated. Despite this "naïve" assumption, Naïve Bayes' algorithm often works well.

Another problem with this approach is that if a word appears in the control set but never appears in Demonstration requirements, then the probability from (2) always computes to exactly 0.0. Laplacian smoothing is used to avoid this situation. With Laplacian smoothing, word frequencies are adjusted by a factor

$$L = (s + \alpha)/(N + d\alpha), \quad (3)$$

where s is the number of word occurrences in the corpus, N is the total number of words in the corpus (including repeats), and α and d are smoothing factors. This study used the values α=1 and d=0. With N=14,370 total words in the control requirements set, this guaranteed that any word X_i had a frequency of at least 1/14,370 =.0000696 in (2).

4.2. Nearest Neighbors by weighted Dice similarity

This algorithm found the control requirement most similar to each validation (Michigan) requirement, where closeness was defined using Dice similarity weighted by Term Frequency-Inverse Document Frequency (TF-IDF) scores. The predicted verification method of the Michigan requirement was then set to the verification method of the nearest neighbor.

The TF of each stemmed word in each requirement was computed for both the control and the validation requirements. Also computed was the document frequencies (DF), which is the counts of the number of requirements that contained each word. From these values, the TF-IDF for each word in each requirement is evaluated as:

$$TF\text{-}IDF(X,D) = f(X,D) * log\left(\frac{1}{f(X,C)}\right) \quad (4)$$

where

- C is a corpus of documents,
- D is a particular document within the corpus C,
- X is a particular word in the document D, and
- $f(X,Y)$ is the frequency of word X amongst all words in the document or corpus Y.

With TF-IDF, a frequently occurring word appearing in only a few documents is weighted heavily, implying that this word has important meaning.

Dice similarity [22] defines the closeness d between two sets S and N by

$$d(S,N) = \frac{2(S \cdot N)}{(|S|+|N|)} \quad (5)$$

where "| |" denotes the number of objects in each set, and "$S \cdot N$" is the raw count of items that S and N have in common. A more general metric, where objects X_i in S and Y_j in N are weighted by positive weighting factors w_i and v_j, is to define the distance between S and N by:

$$d(S,N) = \sum_{X_i = Y_j \in (S \cap N)} \frac{w_i + v_j}{(||S|| + ||N||)} \quad (6)$$

with the norm given by the sum of the weights:

$$||S|| = \sum_{X_i \epsilon S} w_i, \quad ||N|| = \sum_{Y_j \epsilon N} v_j \quad (7)$$

This weighted Dice similarity metric is 0 if *S* and *N* have no common elements, increasing as commonality increases, and evaluates to 1 if they contain exactly the same elements.

The weighted Dice similarity using TF-IDF weights was computed for each requirement paragraph pair, one drawn from the control set (Australia/New Zealand, IAGR and Nova Scotia) and one from the validation set (Michigan). The control set requirement most similar to each validation set requirement was determined, and the VM of the closest control set requirement was used to predict the VM of the validation set requirement.

4.3. Nearest Neighbor with Latent Semantic Analysis

A term-document matrix is a two-dimensional matrix comprised of terms in one dimension (usually rows), document in the other dimension (columns), and a frequency measure in each matrix cell. For example, the frequency measure could be the percentage of words in the document matching the term.

With latent semantic analysis, the term-document matrix is decomposed using singular value decomposition (SVD) and approximated using the most significant eigenvectors. Each document column is then estimated as a linear sum of the resultant eigenvectors, and document "closeness" is computed as the vector dot product between two matrix columns.

Mathematically, let X be the term-document matrix, with terms as rows, documents as columns, and matrix entries as the TF-IDF score for each term in each document. From linear algebra, we know that X has a SVD of

$$X = USV^T \tag{8}$$

where U and V are orthogonal matrices and S is diagonal.

In the LSA algorithm, the k largest eigenvalues in S are selected and used to form a sub-matrix S_k. A sub-matrix U_k is also formed from U by retaining only the rows corresponding to the k largest eigenvalues in S. Similarly, a sub-matrix V_k^T from V^T is formed by retaining the columns corresponding to the columns of the largest eigenvalues in S. The matrix X is then approximated by

$$X_k = U_k S_k V_k^T \tag{9}$$

This approximation X_k to X is made because X may be too large for practical computation.

With this representation, the correlation matrix between columns in X_k is given by

$$\begin{aligned} C_k = X_k^T X_k &= (U_k S_k V_k^T)^T (U_k S_k V_k^T) \\ &= (V_k S_k U_k^T)(U_k S_k V_k^T) \\ &= V_k S_k U_k^T U_k S_k V_k^T \approx V_k S_k S_k V_k^T \end{aligned} \tag{10}$$

where the last expression is approximate because U_k is an approximation of U, and not necessarily strictly orthonormal.

Using this approximate correlation matrix, for any document j, the document "closest" to it is given by document i, where

$$C_k[i,j] = \max\left(C_k[m,j], m \neq j\right) \tag{11}$$

and $C_k[i,j]$ denotes the element in row i and column j of matrix C_k.

An LSA model was created that used a term-document matrix with stemmed words at least three letters long as the terms, requirements as the documents, and TF-IDF scores as the matrix entries. All requirements from all four datasets were included, and the top k=450 eigenvalues were used for the approximation. The requirement in the control set (Australia, IAGR, Nova Scotia) closest to each validation set (Michigan) was selected and the Michigan predicted requirement VM was then set to the VM of that closest requirement.

4.4. Ensemble Model

The last model examined was an ensemble model containing all three of the other three models. Majority vote of the ensemble was used, with predictions randomized when all three models disagreed.

IV. Results and Results Analysis

The Naïve Bayes (NB) model with Laplacian smoothing, Nearest Neighbor model with Dice (NN-Dice) weights, and Nearest Neighbor model with LSA (NN-LSA) weights, and the ensemble model were each used to predict the Michigan VMs. Confusion matrices for each model and each VM are shown in Tables 2-5 below.

Table 1: Michigan VM Results, Naive Bayes

	Predicted			Predicted	
Actual	Not Demo	Demo	Actual	Not Inspection	Inspection
Not Demo	62	17	Not Insp.	149	3
Demo	15	80	Inspection	6	16

	Predicted			Predicted	
Actual	Not Analysis	Analysis	Actual	Not Test	Test
Not Analysis	142	14	Not Test	129	6
Analysis	5	13	Test	14	25

Table 2: Michigan VM Results, NN-Dice

	Predicted			Predicted	
Actual	Not Demo	Demo	Actual	Not Inspection	Inspection
Not Demo	63	16	Not Insp.	144	8
Demo	10	85	Inspection	6	16

	Predicted			Predicted	
Actual	Not Analysis	Analysis	Actual	Not Test	Test
Not Analysis	154	2	Not Test	131	4
Analysis	6	12	Test	8	31

Table 3: Michigan VM Results, NN-LSA

	Predicted			Predicted	
Actual	Not Demo	Demo	Actual	Not Inspection	Inspection
Not Demo	63	16	Not Insp.	144	8
Demo	10	85	Inspection	6	16

	Predicted			Predicted	
Actual	Not Analysis	Analysis	Actual	Not Test	Test
Not Analysis	154	2	Not Test	131	4
Analysis	6	12	Test	8	31

Table 5: Michigan VM Results, Ensemble Model

	Predicted			Predicted	
Actual	Not Demo	Demo	Actual	Not Insp.	Inspection
Not Demo	62.00	17.00	Not Insp.	149.00	3.00
Demo	8.33	86.67	Inspection	6.67	15.33

	Predicted			Predicted	
Actual	Not Analysis	Analysis	Actual	Not Test	Test
Not Analysis	154.00	2.00	Not Test	130.00	5.00
Analysis	4.67	13.33	Test	7.00	32.00

Table 6, below, compares model performance across several factors: sensitivity and specificity by VM, overall accuracy, and the overall sensitivity-specificity geometric mean (SSGM) across all VMs. This latter metric is defined by:

$$SSGM = \prod_{VM_i}(Sensitivity_i * Specificity_i)^{.125} \qquad (12)$$

where VM_i are the verification methods. This metric (12) weights sensitivity and specificity equally, irrespective of the frequency of positives or negatives, and was added because model accuracy sometimes underweights false negatives when positives are rare in the dataset.

We see from Table 6 that the performance of each of the standalone algorithms is comparable to the others across all measures: sensitivity, specificity, accuracy and SSGM, although the NN-Dice performs slightly better. The ensemble model outperformed all of the other models, however, with a model accuracy of 84.7% and SSGM of 85.4%.

Table 6: Comparison of Model Correctness

Metric\Method	NB	NN-Dice	NN-LSA	Ensemble
Demo Sensitivity	84.2%	89.5%	83.2%	91.2%
Demo Specificity	78.5%	79.7%	81.0%	78.5%
Inspection Sensitivity	72.7%	72.7%	63.6%	69.7%
Inspection Specificity	98.0%	94.7%	94.1%	98.0%
Analysis Sensitivity	72.2%	66.7%	61.1%	74.1%
Analysis Specificity	91.0%	98.7%	97.4%	98.7%
Test Sensitivity	64.1%	79.5%	79.5%	82.1%
Test Specificity	95.6%	97.0%	91.9%	96.3%
Overall Accuracy	**77.0%**	**82.8%**	**77.6%**	**84.7%**
Overall SSGM	**81.2%**	**84.1%**	**80.4%**	**85.4%**

Lowest accuracies occurred with NB and NN-LSA. NN-LSA used 450 eigenvectors in the SVD approximation. This result was relatively insensitive to increases in basis size; doubling the basis size to 900 only improved the overall model SSGM from 80.4% to 80.6%, and the overall accuracy from 77.6% to 78.2%.

Two types of errors contributed to the model correctness scores: truth data errors and model errors. This first, an error in the truth data itself, is a human error since truth VMs were manually set using systems engineering expertise. For example, consider the following requirement from [5]:

> (A slot machine) *"must have its logic boards, computer chips, and any other devices that store memory secured in a locked enclosure…*"

This was assigned a truth VM of Inspection. However, the following requirement IAGR regulation below was assigned a truth VM of Demonstration:

> "*Logic boards and memory components for (slot machines) and the access to the logic boards and memory components must be sealed, secured and alarmed*".

These two requirements are similar, but the second includes the phrase "and alarmed". This makes the requirement difficult to verify by inspection alone. Nevertheless, both of the Nearest Neighbor algorithms paired these two requirements, and as a result predicted (incorrectly) that the Michigan VM should be "Demonstration".

Following our rules for assigning truth VMs, the IAGR requirement should have been (manually) factored into two parts to separate out the different VMs.

These human engineering errors were retained in the model results, because the alternative – fixing the data and rerunning the model – could be considered "fitting the model to match the desired result". Hence the analysis results would not have been representative of real-life requirements engineering. Approximately 5% of the verification methods were inconsistently or incorrectly tagged during pre-processing, which happens to be approximately the same as the accuracy range of the standalone models (NB, NN-Dice, NN-LSA).

Overall model accuracy was very good, with standalone models correctly predicting the VM 77%-83% of the time, and the ensemble model correctly predicting the VM 85% of the time (Table 6). As expected, the model errors tended to occur when word contexts differed between the control corpus and the validation corpus. For a striking example of this, consider the following Michigan requirement with a truth VM of Test:

> *"The Random Number Generator and random selection process must be impervious to influences from outside the EGD, including, but not limited to … Electrostatic interference".*

However, the NN-Dice algorithm found the following Australia/New Zealand regulation as its closest match, driven by the matching of the words "select", "limit" and "include":

> *"Double-down rules are to be clearly explained including limitations of which totals may allow a double down to be selected."*

The second requirement, which had a truth VM of Inspection, is significantly different and completely unrelated to its paired requirement. This illustrates a weakness of all of the models considered; they rely on strictly word-based text analyses without additional semantic or syntactic analyses.

V. Summary and Conclusion

Correctly capture stakeholder requirements and their VMs is an essential part of requirements engineering. The VCRM is a mechanism that ensures requirements and test plans are aligned.

Using slot machine legal regulations from Australia/New Zealand, Nova Scotia and Michigan, together with proposed regulations from the IAGR, this paper evaluated four approaches for automatically generating VMs from natural language requirements: Naïve Bayes, Nearest Neighbor using weighted Dice similarity, Nearest Neighbor with LSA weights, and an ensemble approach combining the other three.

Requirements were initially extracted from the regulations using simple heuristics and subsequently reviewed and updated using requirements engineering knowledge. Stop words and punctuation were removed, and the remaining words were stemmed. Using this data, the models were trained on the Australia/New Zealand, Nova Scotia and IAGR data to predict Michigan VMs.

All four models had similar prediction accuracy. Of the standalone algorithms, the NN-Dice model performed the best (82.8% overall accuracy). Better yet was an ensemble model using a majority voting scheme for all three standalone models, which achieved an 84.7% accuracy.

Although these models were good at predicting VMs, they were not good enough to eliminate the role of the requirements engineer. The best model, the ensemble model, had an 85% accuracy meant that 15% of the requirements were mismatched to their VM.

However, when sufficient similar-to requirements data is available, any of these approaches would useful for creating an initial VCRM or discovering inconsistencies in an existing VCRM. About 5% of the truth data VMs were inconsistently tagged during data preparation for this study. As noted in [21], inconsistent VMs can drive up test costs, sometimes considerably.

The algorithms studied in this paper were all word frequency analyses and lacked syntactical or semantical considerations. To improve upon these results, other algorithms that incorporate syntax and/or semantics will most likely be needed. This is the focus of the author's continuing investigations.

REFERENCES

[1] INCOSE (International Council on Systems Engineering). Systems Engineering Handbook – A Guide for System Lifecycle Processes and Activities. John Wiley and Sons, Hoboken, NJ, (2015).

[2] Queensland (Australia) Government, "Australian/New Zealand Gaming Machine National Standard 2015". 2015. Retrieved from https://www.publications.qld.gov.au/dataset/a-nz-gaming-machine-national-standards/resource/5904c4d0-19ea-4769-ae25-fa2605a8c9f2.

[3] Nova Scotia Department of Justice, "Casino Regulations made under Section 127 of the Gaming Control Act". 2020. Retrieved from https://novascotia.ca/just/regulations/regs/ gccasino.htm.

[4] International Association of Gaming Regulators, "International Technical Standards for Electronic Gaming Machines", Version 1.0. 28 October 2014. Retrieved from https://www.iagr.org/membership/international-machine-gaming-technical-standards.

[5] Michigan Gaming Control Board, "Casino Gaming". (n.d.). Retrieved from https://www.michigan.gov/documents/f-rules1_6145_7.pdf".

[6] Liping Zhao, et. al., "Natural Language Processing (NLP) for Requirements Engineering: A Systematic Mapping Study", unpublished. 2020. Retrieved from https://arxiv.org/abs/2004.01099.

[7] F. Mokammel, E. Coatanea, J. Coatanea, E. Blanco and M. Pietola. "Automatic requirements extraction, analysis, and graph representation using an approach derived from computational linguistics." Wiley Online Library, Systems Engineering (2018); 1–21.

[8] J. Natt och Dag, B. Regnell, P. Carlshamre, M. Andersson and J. Karlsson, "A Feasibility Study of Automated Natural Language Requirements Analysis in Market-Driven Development", SpringerLink Requirements Eng (2002) 7:20–33.

[9] K.S. Divya, R. Subha and S. Palaniswami, "Similar Words Identification Using Naive and TF-IDF Method." International Journal of Information Technology and Computer Science, (2014), 11, 42-47.

[10] C.A. Furnari, C. Palomares Bonache, J. Franch Gutiérrez, Orsim: Integrating existing software components to detect similar natural language requirements, CEUR Workshop Proceedings, (2018), pp. 1–7.

[11] N. Niu and S. Easterbrook, "On-Demand Cluster Analysis for Product Line Functional Requirements," 2008 12th International Software Product Line Conference, Limerick, (2008), pp. 87-96, doi: 10.1109/SPLC.2008.11.

[12] Yang, H. and P. Liang. Identification and Classification of Requirements from App User Reviews. EASE'17: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, June (2017), pp. 344–353.

[13] M. Lu and P. Liang, "Automatic Classification of Non-Functional Requirements from Augmented App User Reviews". Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (2017), pp. 344-353.

[14] J. Swadia, A Study of Text Mining Framework for Automated Classification of Software Requirements in Enterprise Systems. Master of Science thesis, Arizona State University. 2016. Retrieved from https://repository.asu.edu/attachments/170770/content/Swadia_asu_0010N_16164.pdf, accessed 10/4/2020.

[15] J. Slankas and L. Williams, "Automated extraction of non-functional requirements in available documentation." 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE), (May 2013), pp. 9-16.

[16] Michael Prendergast, "Automated Extraction and Classification of Slot Machine Requirements from Gaming Regulations", unpublished. 2020. Submitted to 15th Annual IEEE International Systems Conference. Retrieved from https://engrxiv.org/rftwx/.

[17] Anurag Dwarakanath and Shubhashis Sengupta, "Litmus: Generation of Test Cases from Functional Requirements in Natural Language", NLDB'12: Proceedings of the 17th international conference on Applications of Natural Language Processing and Information Systems, (June 2012), pgs. 58–69. https://doi.org/10.1007/978-3-642-31178-9_6.

[18] Satoshi Masuda, Tohru Matsuodani and Kazuhiko Tsuda, "Automatic Generation of Test Cases Using Document Analysis Techniques", International Journal of New Technology and Research (IJNTR) ISSN:2454-4116, Volume-2, Issue-7, (July 2016) pp. 59-64.

[19] Ravishankar Boddu, Lan Guo, Supratik Mukhopadhyay and Bojan Cukic, "RETNA: From Requirements Engineering to Testing in a Natural Way", Proc IEEE Int Requir Eng Conf. 2004 Sep; (2004): 262–271. doi: 10.1109/ICRE.2004.1335683.

[20] Priyanka Kulkarni and Yashada Joglekar, "Generating and Analyzing Test cases from Software Requirements using NLP and Hadoop", International Journal of Current Engineering and Technology; Vol. 4, No. 6 (Dec 2014). Available at http://inpressco.com/category/ijcet.

[21] Baris Guidali, Holger Funke, Stefan Sauer and Gregor Engels, "TORC: Test plan optimization by requirements clustering", Software Quality Journal, Vol. 18 No. 2, (June 2010). doi: 10.1007/s11219-011-9149-4.

[22] L. R. Dice, "Measures of the Amount of Ecologic Association Between Species". Ecology. 26, 3 (1945): 297–302. doi:10.2307/1932409. JSTOR 1932.