Automated Extraction and Classification of Slot Machine Requirements from Gaming Regulations

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

Abstract*—Analyzing stakeholder needs and transforming them into requirements is an important early step in the systems engineering lifecycle [1]. In regulated industries, important technical requirements can be found in state and federal laws and regulations. Casino gaming is one such industry. This paper analyzes South Dakota and Nevada slot machine regulations and applies automated natural language processing to extract and analyze technical requirements derived from them. First, each parts of speech (POS) in the regulations is identified. From this, the important adjective and noun keywords and keyword combinations are extracted using the Rapid Automatic Keyword Extraction (RAKE) algorithm [2]. Next, slot machine requirements are extracted from the gaming laws, many of which lack a "shall" in them. To perform this, a 12-rule pattern matching algorithm that applies phrase substitutions and identifies leader-subordinate paragraph headings is applied to the slot machine gaming rules. This approach successfully extracts nearly all of the slot machine technical and operations requirements, though fails to separate compound requirements accounting for approximately 3% of the total. Then, after stemming and stopping the regulations, a Naïve Bayes model for identifying functional requirements is constructed from the South Dakota regulations and applied to the Nevada regulations. This model is able to predict the Nevada functional product requirements from amongst the full set of extracted requirements with 87.5% accuracy. Finally, using a modified version of the Dice similarity metric where the word counts are weighted by the term frequency-inverse document frequency (TF-IDF) scores, the South Dakota requirements most similar to each of the Nevada requirements is determined. The paired South Dakota and Nevada requirements are then assessed using systems engineering expertise for equivalency and relatedness. Using the geometric mean of sensitivity and specificity as a scoring metric, the pairing algorithm optimum performance is 96.1% accurate in identifying equivalent requirements between the two sets of regulations, and 82.0% accurate in identifying related requirements.

Keywords—requirements analysis, similarity, natural language processing, non-functional requirements, slot machines.

I. Introduction

Stakeholder requirements come from a variety of sources, including contract requirements, market conditions, user feedback, and sometimes, laws and regulations. Laws and regulations are a particularly important source of requirements within well-regulated industries such as aerospace and defense, healthcare, finance, automotive and casino gaming.

This paper uses automated natural language processing (NLP) techniques to extract key concepts and requirements in slot machine gaming regulations, identify which of the extracted requirements are functional (vs. non-functional or operations), and compares slot machine requirements embedded in the gaming laws of different states. An overview of this contribution and previous relevant contributions is found in Section II.

Gaming statutes from South Dakota and Nevada are used for this analysis. Section III summarizes the regulation contents and provides examples. Section IV discusses the use of part of speech (POS) tagging and the Rapid Automatic Keyword Extraction (RAKE) algorithm [2] to identify the most important words and phrases in the regulation sets, forming the basis for a program glossary.

Preparing the data for requirements analysis is described in section V. Not all requirements embedded in state laws have the word "shall" embedded in them. Section V describes a simple 12-rule pattern matching algorithm which successfully extracted over 99% of the requirements from the gaming regulations, although these rules failed to separate out certain types of compound requirements. Automated analyses of the requirements to determine which requirements are functional and which are not is described in section VI, and section VII describes the use of similarity analysis to identify equivalent and related requirements. A summary and opportunities for further research are provided in section VIII.

II. Contribution Summary

A. Previous Contributions

The use of NLP for requirements engineering has been around for some time. Reference [3] provides a good overview of the state of the art for natural language requirements engineering in 2001, and a more up-to-date (2018) overview can be found in [4].

Many of the previous NLP approaches to organize requirements included similarity analysis and/or classification algorithms. Early examples of similarity analysis to group requirements can be found in [5,6]. More recently, similarity analysis has been used to link requirements to design documents

[7]. An application of similarity to derive and categorize app requirements from user reviews can be found in [8,9]. A description of the use of support vector machines to classify a variety of NFR types in electronic health systems and comparison of their performance to Bayesian analysis is found in [10]. A succinct overview of recent work in NFR requirements classification is described in [11], together with examples for mathematical software systems.

Natural language similarity analysis tools useful for requirements analysis are described in [12] and [10]. ORSIM (OpenReq-SIMilarity), a tool that encapsulates several similarity mechanisms, is described in [12], together with results from its application to a test database under the European Union's OpenReq project. NFR locator is a tool that extracts NFRs from free text documents. It and its application to health record management is described in [10].

Previous authors have also used similarity measures to compare requirements sets for reuse opportunities [13] and product line development [14].

B. *This Contribution*

To date, however, none of these techniques have been applied to the casino gaming industry in general, nor used to extract and analyze requirements from gaming laws and regulations in particular. The gaming industry is heavily regulated, and legal statutes for slot machines are an important source of many slot machine product and operations including both functional and non-functional technical requirements. This paper's focus is the use of NLP to extract keywords and requirements from state gaming regulations, characterize the requirements as either functional or otherwise, and discover equivalent and similar requirements between the Nevada and South Dakota gaming regulations.

III. Data Description and Pre-Processing

A. *Data Description*

South Dakota legal statues applying to slot machines are found in South Dakota Codified Law (SDCL), Chapter 20:18:17, "Slot Machine Requirements" [15]. These regulations include many mandated functional and non-functional requirements for slot machines and slot machine networks, as well as operations rules that casinos that offer slot machines must follow. This document also includes traceability information such as the statute number, the source (referencing where the law is documented in the South Dakota Register record), and references to related authorization and implementation bills passed by the South Dakota legislature.

For example, section 20:18:17:41.03 contains the NFR:

"Security levels. The host system must have the ability to structure permission levels and logins…"

On the other hand, 20:18:17:29 is an operations requirement:

"The slot machine drop shall be performed by a minimum of two persons."

The South Dakota regulations have 94 sections, varying in length from a single sentence (such as 20:18:17:41.03 above) to 2 or 3 pages. Altogether, the document contains 649 paragraphs.

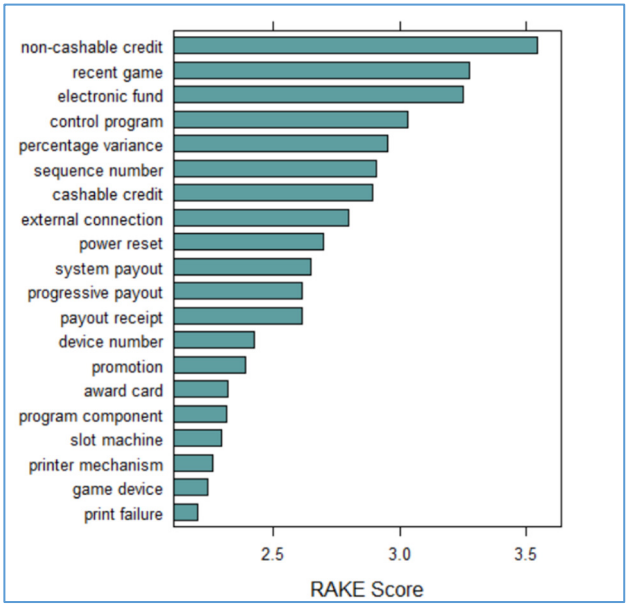

Fig. 1. Keywords and Phrases from Nevada Slot Machine Regulations

The Nevada Gaming Commission (NGC) "Technical Requirements for Slot Machines" [16] includes not only technical requirements but also notes, definitions and annotations indicating the month/year when each regulation was adopted or amended. One functional requirement from this document reads:

"Inappropriate coins-in shall be returned to the player by activation of the hopper or credited toward the next play …"

This NGC document is 15 pages long, and contains 37 sections and 307 paragraphs.

The documents are similar, although the South Dakota regulations contain a few more operations requirements than the Nevada requirements. Automated keyword and key-phrase discovery on the regulations was performed on each dataset. First, automated part-of-speech (POS) tagging was performed on each word in each regulation. Then, the Rapid Automatic Keyword Extraction (RAKE) algorithm [2] was performed on adjective/noun words and combinations. The results from each set of regulations were similar. The top 20 keywords and keyword phrases in the Nevada requirements are shown in Figure 1.

Examining these keywords, one sees that the keywords includes references to states and statuses (print failure, power reset), components (control program, external connection, printer), and slot machine inputs and outputs (electronic fund, cashable credits, non-cashable credits). Hence one might expect that technical requirements dominate these regulations, which turns out to be true.

Gaining stakeholder agreement on key phrases and terms is an essential part of understanding stakeholder needs. To support this, automatically extracted keywords and phrases should be carefully considered when constructing the program glossary. For example, consider the key phrase from Figure 1 "percentage variance". Slot machine networks must be able to compute and report the difference between the theoretical return and the

actual return to players. Stakeholder agreement on how this is calculated is an essential precondition for acceptance testing and certification.

B. Data Pre-Processing

Regulations pre-processing started with identification of all of the paragraphs that contained mandatory and conditional requirements. A conditional requirement is one which may or may not be necessary, depending upon the ultimate system design. An example from [15] of such a requirement is:

"The bonus or extended feature provides only one choice to the patron i.e. press button to spin wheel. In this case the device may auto initiate the bonus or extended feature after a time out period of at least 2 minutes."

If the slot machine manufacturer decides to have an auto-play feature, then this regulation states that this feature may not start until at least 2 minutes have elapsed.

As noted earlier and as seen in the example just given, many of the requirements embedded in legal regulations lack the word "shall". To determine which of these regulations was, in fact, a requirement, the text was transformed using the following substitution rules:

1. "must" or "is required to", "should", "will" → "shall"
2. "may only" → "shall only"
3. "may not" or "is prohibited from" → "shall not"
4. "can" → "may"
5. "[is][are] allowed to" → "may be allowed to"
6. "[is][are] prohibited from" → "shall not"
7. "requires a" → "shall require a"
8. "[is][are] determined" → "shall be determined"
9. "[is][are] responsible" → "shall be responsible"
10. "responsibility … lies" → "responsibility … shall lie"
11. "no … may" → "no … shall"

To this we added a 12th rule: any sub-bullet or sub-paragraph of a requirement statement is also a requirement.

After these transformations, any paragraph or sub-paragraph that contains the word "shall" was deemed to be a mandatory requirement (functional, non-functional or operations), and any paragraph with the word "may" in it contains a conditional requirement.

When a "shall" was included in a main paragraph but not its corresponding sub-paragraphs, the main requirement text was concatenated to each of the subparagraphs and bullets underneath the requirement. For example, from [17], the statement

"*Gaming devices must have electronically stored meters … that record the number of games played:*

a) Since power reset"

b) Since door close"

becomes

"*Gaming devices must have electronically stored meters … that record the number of games played since power reset."*

"*Gaming devices must have electronically stored meters … that record the number of games played since door close."*

The English language is rich and nuanced, and there is no doubt that many more rules would be required to make the list of rules exhaustive. Nevertheless, when these 12 rules were compared with a systems engineering expert opinion review of the gaming regulations, all but one of the regulation requirements were captured. The one requirement not captured states [15]:

"If a slot machine uses more than one similar physical or video component to portray either motion or a random selection process the symbols on each separate component are selected independently and randomly."

Here the phrase "shall be" is implied by the word "are", but the rules listed above are unable to cover this case. This was the only paragraph not recognized to contain a requirement in it across both sets of regulations.

Another limitation of this ruleset is that it fails to separate complex requirements that are partly technical and partly operational. For example, from [16]:

"System meters shall be referred to with the above terms and shall accumulate applicable system generated information as well as information stored on gaming device meters as required by Technical Standard 2.040."

The word "shall" appears twice in this one sentence. One of these is a technical requirement (adherence to 2.040), the other is a documentation requirement. The ruleset used in this analysis was not robust enough to split this sentence up. However, only about 3% of the requirements were compound in this way.

From this ruleset, 273 requirements paragraphs from the Nevada regulations were extracted, and 284 South Dakota requirements paragraphs were extracted.

The final step in pre-processing the gaming regulations was to prepare them for the term frequency and similarity modeling. Stopwords were removed, remaining words were stemmed, punctuation was removed, and words were converted to lower case. For each stemmed word in each requirement, the term frequency (TF) and the word frequency against each regulation

TABLE I. CONFUSION MATRIX FOR CONTROL DATASET

Control Dataset (South Dakota Requirements)		
Actual\Prediction	Non-technical	Technical
Non-technical	128	2
Technical	1	153

TABLE II. CONFUSION MAGTRIX FOR VALIDATION DATASET

Validation Dataset (Nevada Requirements)		
Actual\Prediction	Non-technical	Technical
Non-technical	41	8
Technical	26	198

set was tabulated. These calculated quantities are used in the modeling described in the following sections.

IV. Requirements Modeling

A. Identifying Functional Requirements

After the text transformations described earlier, a term frequency Naïve Bayes model was built from the South Dakota requirements and applied to the Nevada requirements to predict which requirements were functional or not functional.

An Expert Systems Engineering Professional (the author) assessed each of the 284 requirements paragraphs in the South Dakota regulation set and tagged them as either functional (154 of the requirements) or not (130 of them). The requirements that were not functional fell into two categories: nonfunctional product requirements or operations requirements.

Next, the frequency of every word in the South Dakota regulations was calculated, in total and separately for the functional requirements and non-functional/operations requirements. These were then used to develop a Naïve-Bayesian model that was then applied to predict whether or not each of the Nevada requirements was functional or not.

To illustrate how this model worked, let *X* be a word in the South Dakota requirements set, and define the frequency of X in the functional, not functional and complete set of South Dakota requirements as $f_{in}(X)$, $f_{out}(X)$, and $f_{total}(X)$.

From Bayes' theorem, given a randomly selected word *X* and requirement *R*, the probability that a requirement *R* is functional, given that it contains *X*, is given by:

$$P(functional\ R|X\ in\ R) = \frac{P(functional\ R \cap X\ in\ R)}{P(X)} = P(X|functional\ R) * \frac{P(functional\ R)}{P(X)} \quad (1)$$

But all three of the terms in this last expression are known: $f_{in}(X)$ and $f_{total}(X)$ are the word probabilities, and $P(functional\ R)$ is the fraction of South Dakota requirements paragraphs that are functional.

Generalizing this approach, suppose a new requirement being analyzed consists of *n* words, with $m \leq n$ of them found in the South Dakota requirement set. Then if the *m* words are denoted by $X_1, X_2, \ldots, X_m$, the equation

$$P(functional\ R|X_1, X_2, \ldots, X_m\ in\ R) \approx P(functional\ R) \prod_{i=1}^{m} \frac{P(X_i|functional\ R)}{P(X_i)} \quad (2)$$

provides an estimate of the probability that *R* is a functional requirement. The equality is approximate because words within a requirement are not actually independent. Despite this Naïve Bayes' algorithm approach often yields good predictions. The $n - m$ words that are in requirement *R* but not in the South Dakota requirements are ignored in the computation.

One issue with this approach is that if a word appears in only the South Dakota functional requirements but not in the "not functional" requirements set, then this estimate always computes to exactly 1 whenever that word occurs in *R*. A similar situation occurs when the word appears in only the not functional requirements, and the estimate is exactly 0. Laplacian smoothing was used to avoid these situations. In Laplacian smoothing, word frequencies are adjusted by a factor

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

where *s* is the number of occurrences of the word 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*=8,841 total words in the South Dakota requirements set, this guaranteed that any word X_i had a frequency $P(X_i)$ of at least 1/8,841 = .000113 in (2).

The confusion matrix for the control dataset of South Dakota requirements is shown in Table 1 below. Model accuracy was 98.9%. Specificity and sensitivity were 98.5% and 99.4%, respectively. Only three requirements were incorrectly predicted when compared with expert opinion. Two of these were compound with both functional and non-functional parts, and the third described the usage of the on/off switch (which was tagged as functional by the systems engineering expert).

The real test of a model, however, occurs when applied to the validation dataset. The model was applied to the Nevada requirements and compared with expert opinion, which had apriori tagged each requirement as either functional or otherwise. Of the 273 Nevada requirements, 224 had been tagged as functional and 49 were tagged as not functional. Table II shows the confusion matrix of model predictions against the apriori tagging. Model accuracy was 87.5%, sensitivity was 83.7% and specificity was 88.4%.

Of the 26 Nevada regulation paragraphs that the model could not find, 14 were documentation requirements, 5 were requirements embedded in notes or definitions, 4 were references to technical standards, and 3 were mixed paragraphs (containing both technical and non-technical requirements).

The model's high prediction accuracy is due in part to some keywords only appearing in the functional requirements or only appearing in the not functional requirements. For example, the word "transmit" appears in the South Dakota functional requirements, but does not appear in a non-functional requirement. Similarly, the word "advertise" is used in operations requirements only, never in functional requirements.

Another contributor to the model's high prediction accuracy is the degree of commonality between the Nevada and South Dakota regulations, which is discussed in the next section.

V. Finding Similar and Related Requirements

Slot machine vendors need to meet the regulations of all states that they have customers in. Hence, it is important for them to understand the similarities and differences in two sets of sets of state regulations. Term Frequency-Inverse Document Frequency (TF-IDF) and Dice similarity were employed to find requirements equivalency and relatedness between Nevada and South Dakota slot machine laws.

As noted earlier, the TF of each stemmed word in each requirement was computed for both sets of regulations as part of preprocessing. Also computed were the document frequencies (DF), which are the count of the number of requirements that contained each word. From these values, the TF-IDF for each word in each requirement is given by

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

where

- C is a collection ("corpus") of documents,
- D is a particular document in 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 document set Y.

TF-IDF for the same word in different documents will be different, because TF, given by $f(X,D)$ in (4), is different for each document. TF-IDF scores are higher for a word that occurs often but in few documents. The heuristic for this is that a frequently occurring in only a few documents must convey important meaning in the documents that it is used in.

TF-IDF was computed for each stemmed and stopped word in two corpuses, the set of requirements paragraphs extracted from South Dakota and Nevada gaming laws.

First applied to botany [17], Dice proposed a similarity metric between two sets S and N by

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

Where "$S \cdot N$" is the raw count of the number of items S and N have in common, and "| |" is the count of the objects in each set.

In the context of NLP, this approach can be generalized when the words X_i are weighted by positive weighting factors w_i. This modified Dice measure becomes

$$s(S,N) = \sum_{x_i \in (S \cap N)} \frac{2w_i}{(||S||+||N||)}, \quad (6)$$

and the norm is given by the sum of the word weights

$$|S| = \sum_{X \epsilon S} w_i. \quad (7)$$

The modified Dice similarity score equals 1 if S and N have exactly the same words and word frequencies, and 0 if they have no words in common. The closer the word vectors are to having the same words, the higher the score is.

The weighted Dice similarity using TF-IDF weights was computed for each requirement paragraph pair drawn from the Nevada and South Dakota regulations. The most similar South Dakota requirement to each Nevada requirement was selected.

For comparison, expert opinion was used to grade the quality of the pairs selected. A pair of requirements was graded as:

- Equivalent if they were identical, nearly identical, or conveyed the same intent, even if worded differently,
- Related if both address the same capability and the implementation of one was likely to influence the implementation of the other.
- Dissimilar if not equivalent or related.

Figure 2 depicts a box-and-whisker diagram depicting similarity scores for these four categories of requirements.

The weighted Dice similarity was found to be an excellent proxy for requirements equivalence. To generate such a proxy, the metric d' was computed which maximized the geometric mean of model sensitivity and specificity

$$\sqrt{P(score < d' | not\ equivalent)} * \sqrt{P(score \geq d' | equivalent)}.$$

This occurs when d' = .54, the mean metric is .953, and the proxy accuracy is 96.0%. The confusion matrix for this result in shown in Table III.

This approach was also useful, but not as precise, for characterizing requirement relatedness. The maximum metric for this test occurs when d'=.38, the metric mean is .810, and the accuracy of this proxy is 82.1%. The confusion matrix for this situation is also found in Table III.

VI. Summary and Next Steps

A. Summary

Concept definition is usually performed by skilled engineers who use stakeholder information and their domain knowledge to develop solutions that satisfy stakeholder needs. NLP offers promise for helping them do this. This paper demonstrates four such techniques for slot machine development: automated keyword and key-phrase extraction from state slot machine regulations, automated requirements extraction from the regulations, segregation of functional from non-functional and

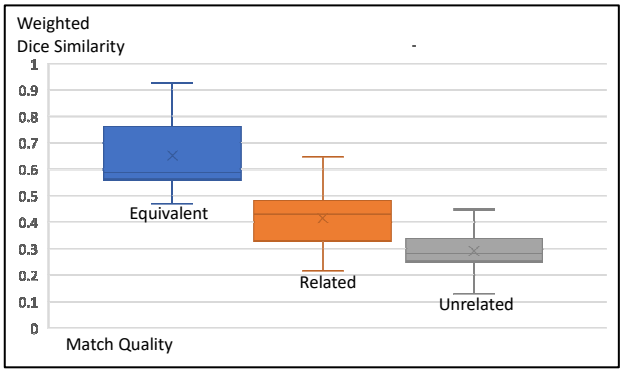

Fig. 2. Match Quality and Modified Dice Similarity

TABLE III. Equivalency and Relatedness Confusion Matrices

Equivalency	Similarity	
	<.54	>=.54
Not Equivalent	245	10
Equivalent	1	17

Relatedness	Similarity	
	<.38	>=.38
Not Related	131	14
Related+Equivalent	35	93

operations requirements, and identification of similar and related requirements across different sets of state gaming laws.

Using statutes regulating the technical standards for slot machines from Nevada and South Dakota, the most important keywords and key phrases were extracted using the RAKE algorithm. These form the basis for the initial program glossary; getting a common understanding of the most important words and phrases is essential to any system development project.

Not all customer needs are written with "shall" statements. Other phrases, such as "must", and "may not" are often used by stakeholders to identify their most important needs. This is especially true in legal statutes. This paper illustrates how a pattern matching algorithm with 12 rules was able to identify extract all but one (99.8%) of the requirements from the Nevada and South Dakota gaming statutes.

After requirements are identified, they are characterized and organized. Using a Naïve Bayes model built from word frequencies in the South Dakota regulations statutes, 88% of the functional requirements and 84% of the non-functional requirements in the Nevada statutes were correctly identified.

Finding commonality and differences across multiple sets of state regulations is important for vendors who want to sell their product across multiple jurisdictions. Using a modified Dice similarity that employs TF-IDF weights, equivalent requirements were identified between the two-state slot machine regulation datasets with 96% accuracy, and related requirements were identified with 82% accuracy.

B. Next Steps

This study focused on slot machine requirements and gaming regulations. There are many other domains that have technical requirements embedded in legal statutes, including but not limited to, the defense industry, the drug and healthcare industry and the automobile industry. Similar undertakings can be performed in these fields.

There are many additional opportunities for further research into the use of NLP for electronic gambling systems, including but not limited to automated architecture creation, automated execution timeline development and automated requirements decomposition. In addition, the results of this study can be further refined and improved upon through the use of more sophisticated algorithms.

Other means to improve this study include: a) adding additional sets of regulations, including the European Union regulations, and b) developing a more refined approach for requirements extraction, including breaking apart compound requirements that have both functional and non-functional elements, and c) characterizing the NFR requirements more precisely (e.g., security NFR requirements, reliability NFR requirements and maintainability NFR requirements). Potential approach avenues for doing this can be found in [8, 9, 10, 11]. This is a direction of ongoing research by the author.

Although this paper illustrated how to estimate the similarity between two paragraphs, it did not cluster sets of requirements by similarity. The use of clustering algorithms for grouping slot machine and other casino requirements is also a direction of ongoing research by the author.

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] S. Rose, D. Engel, N. Cramer and W. Cowley, "Automatic Keyword Extraction from Individual Documents". Chapter 1, Text Mining: Applications and Theory, edited by Michael W. Berry and Jacob Kogan © 2010, John Wiley & Sons, Ltd.

[3] D. M. Berry, "Natural Language and Requirements Engineering — Nu?", International Workshop on Requirements Engineering, Imperial College, London, UK, 2001. Retrieved from https://files.ifi.uzh.ch/rerg/arvo/IWRE/papers%26presentations/Berry.pdf, accessed 10/4/2020.

[4] 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.

[5] 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.

[6] J. Natt och Dag, B. Regnell, V. Gervasi and S. Brinkkemper, "A linguistic-engineering approach to large-scale requirements management," in IEEE Software, vol. 22, no. 1, pp. 32-39, Jan.-Feb. 2005, doi: 10.1109/MS.2005.1.

[7] 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.

[8] 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 Pages 344–353. .

[9] 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, pgs. 344-353.

[10] 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), pages 9–16, May 2013.

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

[12] 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.

[13] E. J. Stierna and N. C. Row, "Applying information-retrieval methods to software reuse: a case study." Information Processing and Management, Vol. 39, No. 1 (January 2003), 67-74.

[14] 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.

[15] South Dakota Legislature, "Slot Machine Requirements", South Dakota Codified Law (SDCL) Chapter 20:18:17. Retrieved from https://sdlegislature.gov/, accessed 10/4/2020.

[16] Nevada Gaming Commission – Technology Division (NGCTD). (2005). Technical Standards for Gaming Devices and On-line Slot Systems. Retrieved from https://gaming.nv.gov/, accessed 10/4/2020.

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