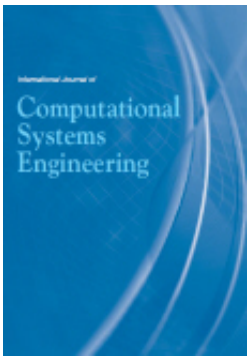


International Journal of Computational Systems Engineering

ISSN online: 2046-3405 - ISSN print: 2046-3391
https://www.inderscience.com/ijcsyse

Algorithmic and meta-algorithmic machine learning natural language processing approaches for stakeholder requirements classification

Arturo N. Villanueva Jr., Steven J. Simske

DOI: 10.1504/IJCSYSE.2023.10056319

Article History:

Received:	19 May 2022
Last revised:	04 April 2023
Accepted:	04 April 2023
Published online:	19 May 2023

Algorithmic and meta-algorithmic machine learning natural language processing approaches for stakeholder requirements classification

Arturo N. Villanueva Jr.* and Steven J. Simske

Colorado State University,
Fort Collins, Colorado, USA
Email: art.villanueva@colostate.edu
Email: steve.simske@colostate.edu
*Corresponding author

Abstract: Requirements engineering begins with discovery at the outset of project acquisition. Documents typically used during this phase include statements of works (SOWs) and requests for proposals (RFPs). One of the first challenges of a systems engineer is to carefully classify requirements into appropriate bins for further processing. This manual process, fundamental to understanding stakeholder needs and architecting and designing the system(s) of interest, is often tedious, particularly for large projects that start out with thousands of requirements embedded in these documents, making the task ripe for automation. For this research, we investigate multiple combinations of algorithms and meta-algorithms to glean insight as to how well they perform on this aspect of one of the more mundane aspects of requirements engineering. We obtain, by running various training corpora representing multiple industries through our pipelines of (meta-)algorithms, some understanding of what works best and what and how they could be improved.

Keywords: natural language processing; NLP; classification; meta-algorithms; machine learning; statements of works; SOWs; requests for proposals; RFPs; document classification.

Reference to this paper should be made as follows: Villanueva Jr., A.N. and Simske, S.J. (2022) 'Algorithmic and meta-algorithmic machine learning natural language processing approaches for stakeholder requirements classification', *Int. J. Computational Systems Engineering*, Vol. 7, No. 1, pp.41–56.

Biographical notes: Arturo N. Villanueva Jr. is a Doctor of Engineering student in Systems Engineering at the Colorado State University. He is also the AI/ML Chief Technology Architect for the Federal Division of Dell Technologies. He has served as lead systems engineer for multiple high-profile programs, including billion-dollar initiatives. Art is an entrepreneur, having founded two renewable energy start-ups, one of which is now trading on the NASDAQ stock market; and an inventor with three patents. He obtained his Master's in Architecture-based Enterprise Systems Engineering from UCSD and his Bachelor's in Applied Mathematics with a specialisation in Computing from UCLA.

Steven J. Simske is a Professor in Systems Engineering, and affiliate of Biomedical and Mechanical Engineering at the Colorado State University (CSU). He is an author of 225 US patents and more than 450 publications, he is an IS&T, IEEE, and NAI Fellow. He is the Steering Committee Chair for the ACM DocEng Symposium, and past member of the World Economic Forum Global Agenda Councils for Illicit Trade, Illicit Economy and the Future of Electronics. At CSU, he has a cadre of on-campus students in his engineering departments, along with a larger contingent of on-line/remote graduate students researching in a wide variety of disciplines.

1 Introduction

Requirements engineering (RE), the discipline within systems engineering dealing with the development, analysis, and management of requirements that define a system at successive levels of abstraction (Dick et al., 2017), can be laborious. One of the most tedious aspects of this area commences at the onset of the project, or even before, with the contractor's receipt of a requests for proposal (RFP) or statement of work (SOW) [Sainani et al. (2020) for example, report experts classifying a mean of 17 requirements per hour by hand]. While small projects may consist of only a few top-level requirements, large enterprise-scale endeavours such as a public transportation system or a new communications infrastructure for the US Navy's fleet may have thousands. According to Jones (2000), requirements development alone constitute 7.0% of a project's cost for commercial projects and 10% for military software, translating to 22.7 and 17.5 person-months in requirements development, respectively.

One such tedium in RE is the classifying of requirements, which is particularly important in the beginning, for various reasons. Various ways of classifying have been proposed and used for diverse applications:

1 Classification by contract obligations, whether governance or architectural (Sainani et al., 2020).

2 Classification by hierarchy or detail (INCOSE, 2015).

3 Classification by functionality or non-functionality (Glinz, 2020).

4 Classification by types (functional, performance, design constraints, quality attributes) (Hefner, 2019).

5 Classification by quality attribute and expertise needed (cybersecurity, reliability, etc.) (Clements et al., 2003; Rozanski and Woods, 2011).

6 Classification by importance or urgency (Dick et al., 2017).

The systems engineering 'Vee' model, regarded as the standard process for systems engineering (INCOSE, 2017), starts out on the left side to address requirements decomposition starting from stakeholder needs and requirements. This paper describes, via machine learning and meta-algorithmic patterns, one of the Vee model's earliest needs for organisation by taking as input, customer-provided SOWs written in natural language and classifying requirements as governance or system requirements, similar to #1 above. We, however, did not want to separate between contract obligations into governance and architectural, because architecture has a narrower meaning (Rozanski and Woods, 2011) in the context of a solution. Design constraints and functional requirements (FRs) need to also be considered. As such, we opted to separate statements into the following:

1 systems requirements, those that are levied on the system being developed and delivered, including FRs and non-functional requirements (NFRs) (constraints, performance, and quality attributes)

2 governance requirements, which are requirements that are not system requirements, and instead are those that are levied on the project team and other support, and includes project delivery, compliance, execution, training, operation, maintenance, and other services.

This separation is often necessary so that the project management and support teams can focus on the support activities and the engineering team and focus on the technical areas. Such a dichotomy is often not cut and dried, however, as we will see, as some requirements straddle the line. Requirements such as "the contractor shall produce the information model of the system and its components" are indeed levied on the contractor yet are targeted towards the engineering team; and "the contractor shall implement a zero-trust architecture" appears to be levied on the contractor but actually describes a constraint requirement levied on the system. The training corpora reflect the binary classification bins as two documents are used – a document that is purely programmatic (such as a performance work statement, or PWS) and a document that describes a system (DAU, n.d.), such as a specification.

For the purposes of this research, we adhere to standard contract language (Black's Law Dictionary Free Online Legal Dictionary, n.d.) and universally accepted convention that requirements are identified to have the imperative 'shall' following the subject (INCOSE, 2015; DAU, n.d.; Douglass, 2016; Anon., 2018; NASA, n.d.), loosely of the form "<subject> shall <action verb clause> <object clause> <optional qualifying clause>." Some examples include "the vendor shall provide a monthly-updated integrated master schedule within 30 days of award date", "the system shall be accessible as per the Americans with Disabilities Act (ADA)", and "the contractor shall utilise model-based systems engineering (MBSE) principles." With the convention, we distinguish requirements from needs, which are typically not written in the 'shall' structure. Capabilities, operational and mission threads, needs, use cases, and user stories may be used to derive requirements but do not qualify as requirements themselves. A formal conversion to the 'shall' structure is necessary not only for consistency with a standard form, but also to vet the stakeholders' wants and expectations (INCOSE, 2022).

2 Related work

In recent years, some work has been done in classifying requirements using machine learning techniques, though many have focused on software engineering projects. There is notable work on the subject:

Sainani et al. (2020) started with 20 software contracts, extracted obligations (requirements) from them, and classified those obligations using Naïve Bayes, random forest, and support machine vectors (SVMs), as well as using a bidirectional long-short-term memory (BiLSTM) deep learning method and Google's BERT for comparison. Similarly, Canedo and Mendes (2020) studied multiple algorithms [logistic regression (LR), SVMs, multinomial Naïve Bayes (MNB), and k-nearest neighbours (kNN)] for their accuracy and precision for classifying. Earlier work by Mahmoud and Williams (2016) used word similarity and clustering techniques.

Abad et al. (2017) looked at software requirements from software requirements specifications (SRSs) and classified them into FRs and NFRs. They investigated the effect of pre-processing the dataset by applying grammatical, temporal, and sentimental characteristics of sentences using parts of speech (POS) tagging to standardise the dataset requirements for simpler processing. Finally, they classified NFRs into quality attributes using multiple algorithms, with Naïve Bayes taking the trophy.

Sabir et al. (2020) tackle misclassification of NFRs by assigning multiple tags to requirements with the premise that requirements often straddle a grey area when it comes to correctly categorising them.

Giannakopoulou et al. (2020) describe FRETISH and Lucio et al. (2017) describe EARS, structured natural languages for formally writing requirements, useful for reducing conflicting interpretations and improving analysis.

Besides the work by Sainani et al. however, not a lot has been conducted on ingesting raw SOW data and splitting them into a set that needs to be consumed by project management and a set that needs to be consumed by the systems engineering and technical team. The work led by Sainani focused on investigating software (not complex systems) contracts. What they considered as architectural contract obligations were somewhat limited to architectural constructs specific to software. Complex systems, on the other hand, are a generalised collection of interconnected and interrelated parts, and add another dimension beyond bits and bytes (Ladyman and Wiesner, 2020). Contractual obligations for software projects as these systems typically include uniquely include physical quality attributes such as reliability and availability, and constraints such as size, weight, and power (SWaP) (Kossiakoff et al., 2011).

3 Context and research goals

3.1 Context

The general research area is deemed a natural application of machine learning and automation (Kotonya and Sommerville, 1998), as the vast majority of RFPs and SOWs do not follow a universal structured natural language such as EARS (Mavin et al., 2009) or FRETISH (Giannakopoulou et al., 2020), or at least follow a semi-restricted format that begins with either 'the system shall' or 'the contractor shall'. Such restrictions are unlikely to be enforced by every SOW writer. The PROMISE database itself has only 455 of its 604, or roughly 0.75 (Cleland-Huang, 2007), requirements compliant with the latter restrictions.

We do make at least one assumption: for a statement to be called a requirement, it must contain a 'shall' as per what is mentioned in the introduction above. Other keywords could be used including 'will' or 'should', but following standard RE practices (Anon., 2018; DAU, n.d.; INCOSE, 2017), we stick with 'shall'. In this scheme, requirements that have bullets or lists such as "the system shall comply with (a) standard A, (b) standard B, and (c) standard C" counts as a single requirement. This, of course, violates the rule that requirements need to be singular (Anon., 2018).

3.2 Research goals

Given the above context, we endeavour to discover how some fundamental algorithms (Naïve Bayes, TF*IDF with cosine similarity, and logistical regression) compare with each other as well against a simple pattern match and a more complex unsupervised learning algorithm in Stanford's GloVe (Pennington et al., 2014). The first three will also be subjected to two first order meta-algorithms (Simske, 2013) in the form of weighted voting and predictive selection.

In addition, using GloVe we determine the effect of various parameters on classification accuracy, both as a standalone algorithm and also as the initial categoriser for predictive selection. In particular, we use the most prevalent words found in the training corpora, starting with the single most common word for the governance bin and similarly for the system bin, and increasing number of words to the second most common, and so on until 15 of the most common words (Table 4) are represented.

4 Process/tasks

We executed the following process tasks:

4.1 Data collection

The process of collecting appropriate datasets for analysis was long and arduous, mostly for the lack of publicly accessible ground-truthed requirements sets from which to train and test our engines. The vast majority of publicly-available SOWs either do not come with ground truth labels, are limited to software, small (under 50 requirements), heavily governance-based statements, or a combination of some or all of these. To get around this, for training purposes, we settled on substituting five reasonably large, expired SOWs and specifications, each containing nothing but governance statements or system statements, specifications, and technical descriptions. Because some corpora, such as specification documents, do not have many 'shall' statements but are otherwise relevant to training because of the vocabulary used for those corpora, the numbers of words (and their frequencies) were deemed more relevant than the actual number of "shall" statements. For completeness, however, these numbers are also included for reference (Table 1).

All 21 combinations with at least one governance corpus and one system corpus were used for training. For reference, see Table 2 for the various combinations and their IDs.

For the test phases for each algorithm, despite the tedium of ground-truthing, we had little choice but to do so. We collected and manually processed a set of 13 SOW corpora (Table 3).

Extraction of the 'shall' statements was trivial. Classifying them as either governance or system requirements for ground-truthing was time-consuming but provided some insight: note that some engineering-related requirements such as requiring the use of model-based systems engineering (MBSE) were best classified as governance requirements as these are levied on engineers rather than the system in context. In addition, it was decided that given a choice between more false positives in the classification of system requirements and more false positives in the classification of governance requirements, the former was more tolerable, as some governance requirements could often be better satisfied by system capabilities. For example, if the SOW required the

contractor to provide weekly data dumps from the system, it might be best to implement a feature in the system to automatically send the required data automatically at the required intervals. Or if training was required, a system that was developed to be user-friendly would reduce the amount of manual training involved.

Table 1 Training corpora

Type	*Corpus*	*Industry/type*	*'Shall' statement count*	*Word count*
Governance	TrainG01	Defence/ enterprise communications network	696	163,437
Governance	TrainG09	Construction/ laboratory	199	81,235
System	TrainS02	Agriculture/ conservation management system	8	55,369
System	TrainS05	Defence/ communications and data management system	6,251	668,912
System	TrainS12	Defence/ electronic warfare system	910	200,622

Four critical infrastructure sectors are represented – transportation, defence industrial base, energy, and commercial facilities – each of whose representative SOWs weigh more heavily towards governance-related requirements.

4.2 Preparation and pre-processing

Pre-processing the training datasets involved the following:

- A typical first step, every word in the corpora was converted to lowercase.
- A total of 337 stop words were removed starting with Scikit-Learn's (Pedregosa et al., 2011) English stop words combined with the words *annex*, *appendix*, *diagram*, *example*, *fig*, *figure*, *handbook*, *may*, *mil*, *must*, *page*, *requirement*, *shall*, *table*, *unless*, *use*, *used*, *will*, *within*, and *would*. These additional common words, a few corpus-specific words (*steward* and *watershed*) and the name of the project proved to be prevalent and added no value to the corpora. We used Scikit-Learn's set of stop words as our basis as it was one of the most extensive sets to remove the most common words in the English language.
- Punctuation marks were removed.
- Next, words were combined to prepare for counting by passing all of them through NLTK's (Bird et al., 2009) lemmatisation function three times, first treating everything as nouns, followed by verbs, then adjectives.
- The final step involved removing all words that were not in NLTK's set of 235,892 English words.

Table 2 Training/validation corpora combinations

Training ID	*Validation ID*	*Corpora combination*
TR01	V01	TrainG01 + TrainS02
TR02	V02	TrainG01 + TrainS05
TR03	V03	TrainG01 + TrainS12
TR04	V04	TrainG01 + TrainS02 + TrainS05
TR05	V05	TrainG01 + TrainS02 + TrainS12
TR06	V06	TrainG01 + TrainS05 + TrainS12
TR07	V07	TrainG01 + TrainS02 + TrainS05 + TrainS12
TR08	V08	TrainG09 + TrainS02
TR09	V09	TrainG09 + TrainS05
TR10	V10	TrainG09 + TrainS12
TR11	V11	TrainG09 + TrainS02 + TrainS05
TR12	V12	TrainG09 + TrainS02 + TrainS12
TR13	V13	TrainG09 + TrainS05 + TrainS12
TR14	V14	TrainG09 + TrainS02 + TrainS05 + TrainS12
TR15	V15	TrainG01 + TrainG09 + TrainS02
TR16	V16	TrainG01 + TrainG09 + TrainS05
TR17	V17	TrainG01 + TrainG09 + TrainS12
TR18	V18	TrainG01 + TrainG09 + TrainS02 + TrainS05
TR19	V19	TrainG01 + TrainG09 + TrainS02 + TrainS12
TR20	V20	TrainG01 + TrainG09 + TrainS05 + TrainS12
TR21	V21	TrainG01 + TrainG09 + TrainS02 + TrainS05 + TrainS12

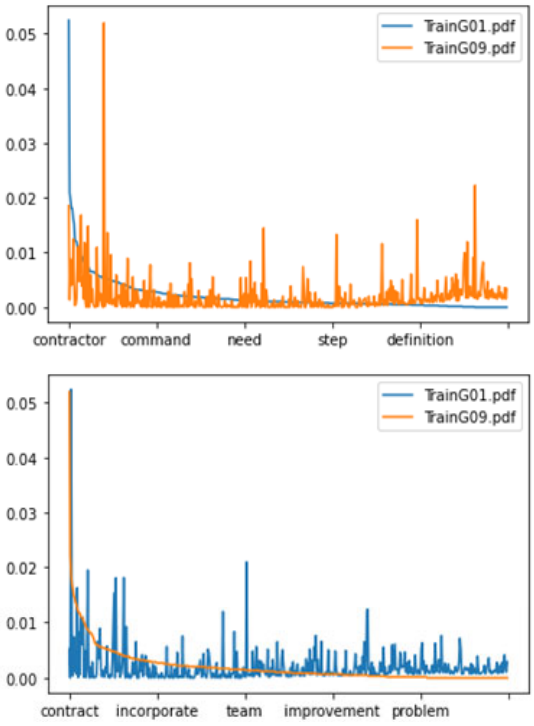
Table 3 Test corpora

SOW #	*Industry/type*	*Requirement counts*		
		Governance	*System*	*Total*
Test01	Transportation system	23	15	38
Test02	Transportation services	19	0	19
Test03	Defence/C4ISR installation services	381	8	389
Test04	Energy/solar system	331	25	356
Test05	Defence/ communications system	53	0	53

Table 3 Test corpora (continued)

SOW #	*Industry/type*	*Requirement counts*		
		Governance	*System*	*Total*
Test06	Defence/inventory mgmt. system prototype	50	11	61
Test07	Defence/ cybersecurity services	40	2	42
Test08	Civil engineering/ construction	174	71	245
Test09	Defence/ communications system	141	4	145
Test10	Defence/ application infrastructure	9	15	24
Test11	Energy/microgrid	8	55	63
Test12	Defence/ cybersecurity system (software)	20	314	334
Test13	Defence/ communications system	11	315	326
	Total	1,260	835	2,095

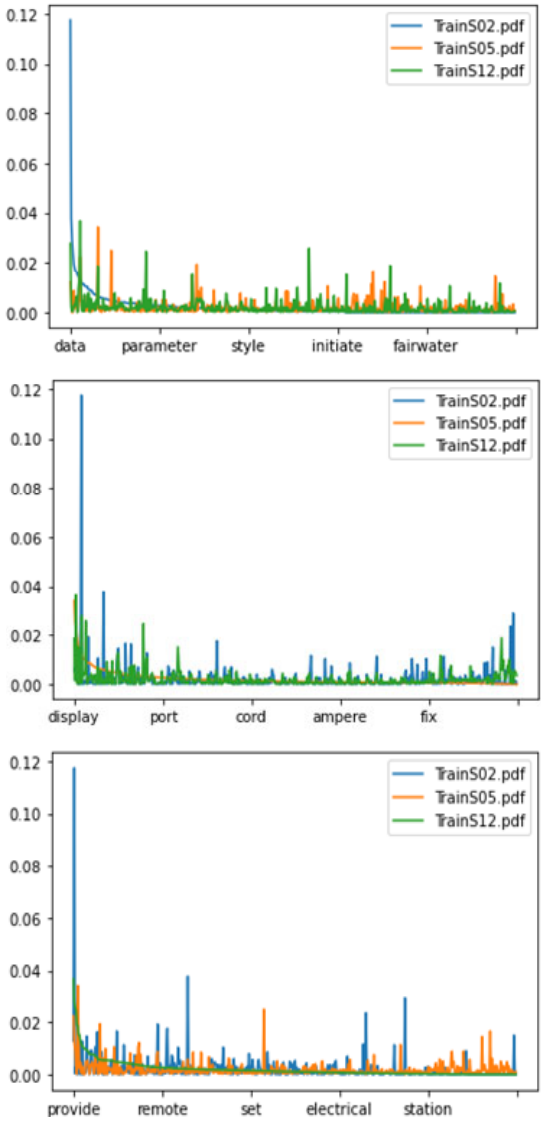
Figure 1 Probability distribution showing Zipf's law normalised for the two training corpora representing governance (see online version for colours)



4.3 Vectorisation

The next step was vectorisation. Using Scikit-Learn's CountVectoriser with a maximum of 500 features and a minimum of one mention, we created vectors for each of the documents or combined set of documents that represented a classification (governance or system). Table 4 shows the probability distribution for the first 15 most common words associated with each of the classification corpora and Figures 1 and 2 show the complete results of the normalised vectors following Zipf's law. The resulting vectors were then used for both Naïve Bayes and TF*IDF/cosine similarity training. Note that training (and later validation for GloVe) used the documents or combination of documents as a whole to generate the vectors since some documents, such as specification documents, do not have 'shall' statements but still contained valuable information on the types of words that is useful for classifying requirements statements.

Figure 2 Probability distribution showing Zipf's law normalised for the three training corpora representing system (see online version for colours)



5 Analysis and results

The comprehensive set of results, using TR21/V21 are depicted in Figures 3 and 4. The other training/validation

corporate resulted in similar results. What is interesting to note here is the enormous variation among the GloVe variants.

We dive deeper into the results in the following subsections.

5.1 Traditional algorithms

5.1.1 Simple pattern match

In order have a good idea on how well our chosen machine learning natural language processing (NLP) algorithms work, it is useful to set an extremely simple baseline. For our test dataset, we simply checked to see if certain words existed in a statement. For the governance classification, we picked the words *contractor*, *vendor*, *offeror*, *provide* and *support*. Statements that did not have these keywords were classified as system. As expected, performance was poor (training accuracy = 0.687) and can partially be attributed to SOW diversity, such as substituting the actual name of the vendor for the terms *vendor* or *contractor*.

Figure 3 Accuracy distribution of classification using all 40 algorithms and meta-algorithms over 13 test corpora for the TR21 training corpora/V21 validation corpora (see online version for colours)

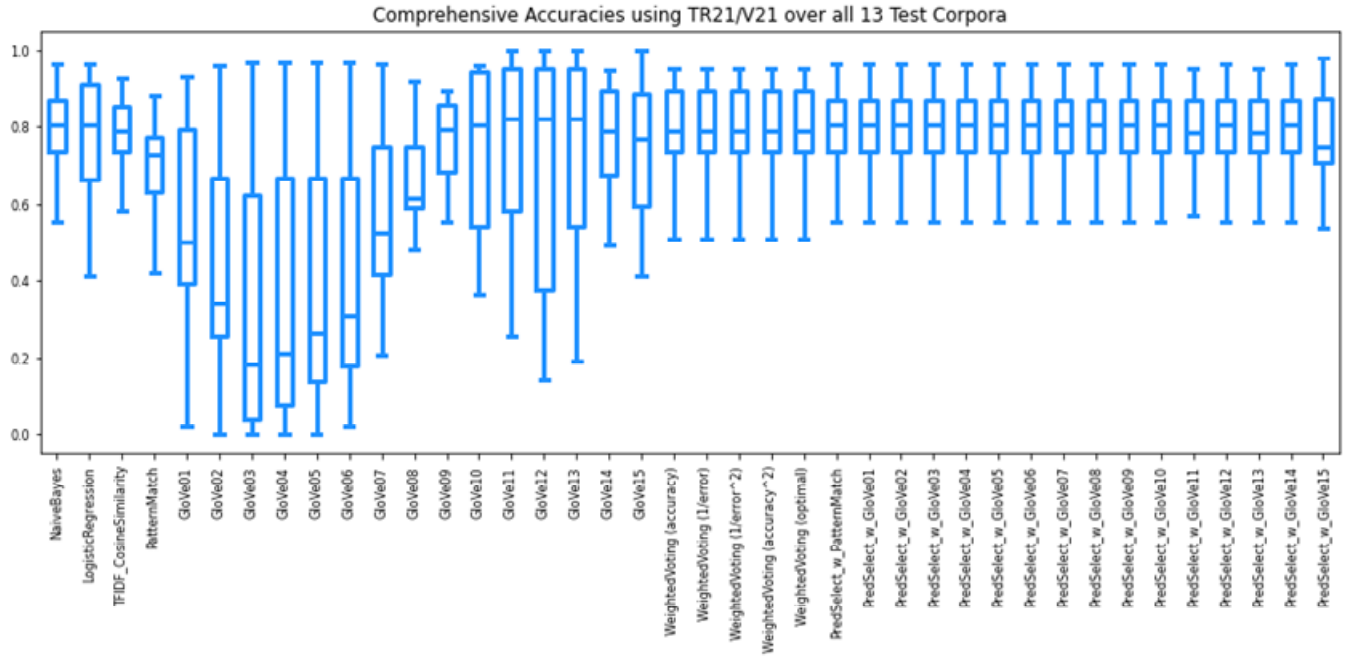


Table 4 Top 15 word probabilities for training corpora

Governance training corpora				*System training corpora*					
TrainG01.pdf		*TrainG09.pdf*		*TrainS02.pdf*		*TrainS05.pdf*		*TrainS12.pdf*	
contractor	0.0593	contract	0.0667	data	0.1129	display	0.0352	provide	0.0367
test	0.0237	construction	0.0286	user	0.0364	section	0.0256	data	0.0277
provide	0.0221	contractor	0.0239	search	0.0281	provide	0.0229	operator	0.0257
support	0.0206	service	0.0217	site	0.0226	function	0.0199	capability	0.0245
plan	0.0205	officer	0.0207	component	0.0185	trim	0.0170	track	0.0188
government	0.0185	review	0.0192	specification	0.0169	control	0.0152	display	0.0185
report	0.0174	draw	0.0187	access	0.0159	drain	0.0148	control	0.0152
technical	0.0139	require	0.0176	design	0.0156	accordance	0.0125	channel	0.0151
train	0.0136	project	0.0172	model	0.0143	data	0.0123	interface	0.0127
management	0.0134	government	0.0159	time	0.0140	alarm	0.0116	distribution	0.0116
include	0.0123	clause	0.0153	provide	0.0124	indication	0.0108	support	0.0108
design	0.0122	work	0.0151	interface	0.0124	mode	0.0108	increment	0.0106
follow	0.0107	business	0.0148	support	0.0121	test	0.0102	print	0.0099
engineer	0.0106	design	0.0142	server	0.0112	operator	0.0099	revision	0.0097
service	0.0106	document	0.0140	management	0.0108	refer	0.0091	equipment	0.0095

Figure 4 Accuracy distribution of classification using all 40 algorithms and meta-algorithms over 13 test corpora for each training corpus combination (see online version for colours)

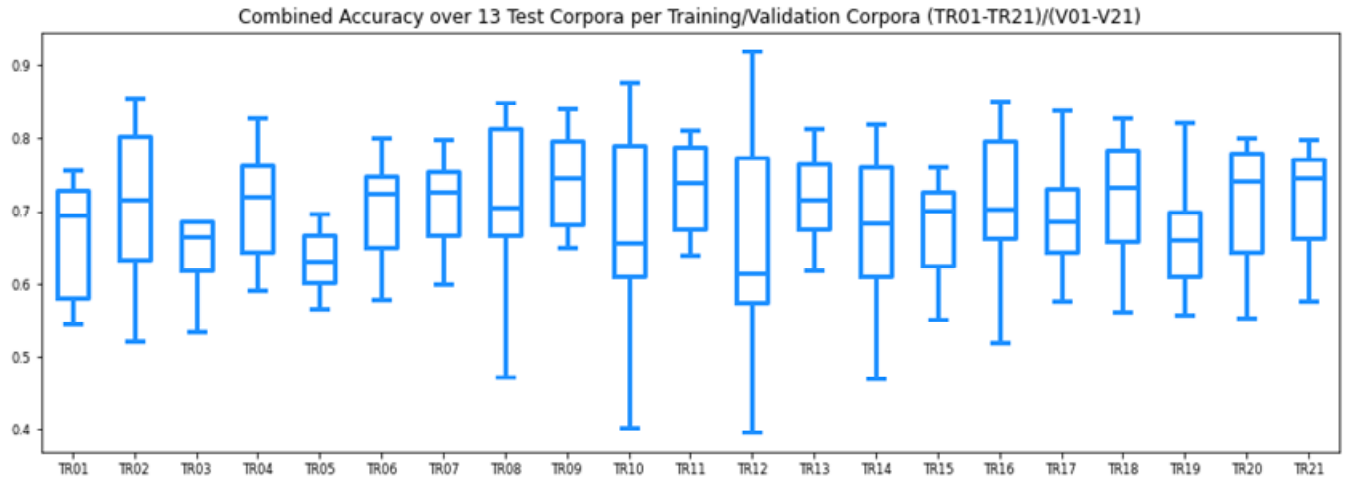


Figure 5 Comparing basic algorithms (pattern match, Naïve Bayes, TF*IDF/cosine similarity, and LR) accuracies for training corpora (TR01-TR21) (see online version for colours)

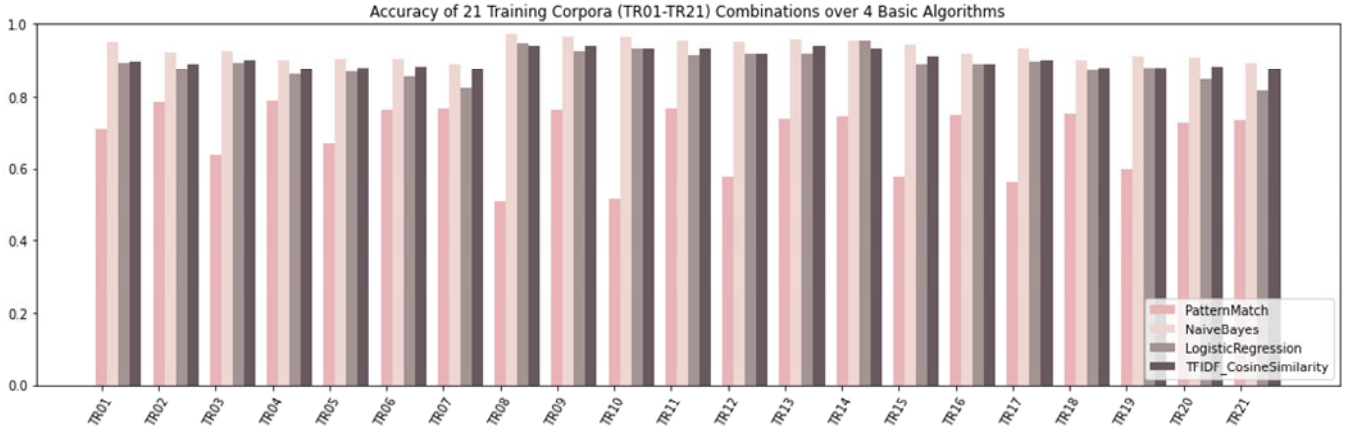
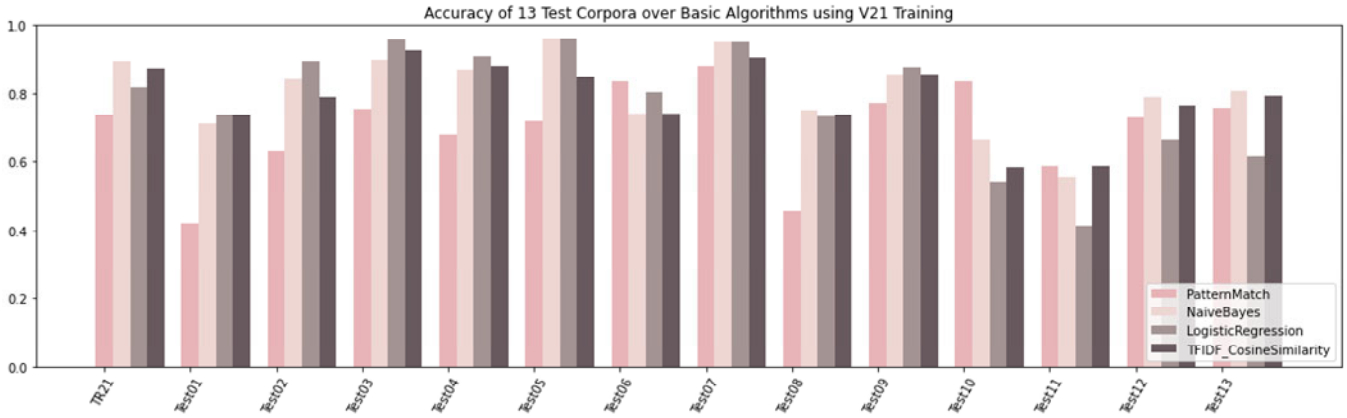


Figure 6 Comparing basic algorithms (pattern match, Naïve Bayes, TF*IDF/cosine similarity, and LR) accuracies for test corpora (TR01-TR13) using TR21 training corpora (see online version for colours)



The next step was to compare accuracy with the results taken from three fundamental NLP classification methods.

5.1.2 Naïve Bayes

Using the vectorised documents described above, our first attempt using the NLTK implementation appeared to be ultra-sensitive to the small training dataset and produced unusable results. The second attempt involved going back to basics for an implementation (Lane et al., 2019) that resulted in much more productive classifications. For this run, we obtained a mean accuracy of 0.929 across all the training corpora.

5.1.3 TF*IDF/cosine similarity

Again, using the vectorised documents, we employed term frequency – inverse document frequency (TF*IDF), calculating the weighted cosine similarity between the requirements and each of the two classification documents. For this run, we obtained a mean accuracy of 0.903 across all training corpora.

5.1.4 Logistic regression

Finally, of the traditional algorithms, we applied LR. For the gradient descent portion of this algorithm, we used $\propto = 10^{-8}$ and limited iterations to 500. For this run, we obtained a mean accuracy of 0.890 across all the training corpora.

All three NLP ML algorithms performed considerably better than the simple pattern match and also shows slightly better results for both Naïve Bayes and TF*IDF/cosine similarity compared to LR. It exhibits the expected comparative accuracy between the two classifiers that use the same word vectors as a basis.

Results of these first four algorithms using all training corpora TR01-TR21 are summarised on Figure 5.

5.1.5 GloVe

Next, we implemented a more sophisticated algorithm using Stanford's GloVe. The algorithm takes advantage of global distances between words using word embeddings or multi-dimensional vector representations. Unlike with using CountVectoriser, GloVe word representations may number in the hundreds of dimensions. Our first implementation created a model from the training corpora described above, but resulted in a low accuracy of <0.60. With the lack of large requirements-specific ground-truthed documents, we looked to use a 100-dimensional word representation database based on a combination of the entirety of Wikipedia from 2014 and the Fifth Edition of the English Gigaword (Pennington et al., 2014; Parker et al., 2011).

This second run used the words in Table 4 as anchor words to measure proximity to the bins and consisted of 15 sub-runs, each run corresponding to the number of words from most common to least common. For example, in the TrainG01 and TrainS02 combination, GloVe1 was fed the words *contractor* and *data* to represent the governance and system bins respectively. In the TrainG09 and TrainS05 combination, GloVe3 was fed the words *contract*, *construction* and *contractor* to represent the governance bin, and *display*, *section* and *provide* to represent the system bin. Combined documents used the combined normalised word probabilities of those documents, with Figure 8 detailing the results of the GloVe variations on the different test corpora. Figure 5 includes the best validation values of the GloVe variations in Figure 8, which were trained with the Wikipedia + Gigaword combination.

Figures 7 and 8 suggest GloVe is highly sensitive to the number of words used as anchors, and generally peaked in accuracy (for the validation corpora) using the top ranked 6–8 common words and dropped again as more words are used. Yet, GloVe09 performed the best and most consistent results with the test corpora. For some test corpora, GloVe performed extremely well, but it could likely be attributed to a bias of the system-oriented nature of those corpora (Table 3). In general, the best performing GloVe variations did not perform any better than the traditional algorithms and consistency of performance was rather undependable (see Figures 9 and 10).

5.2 Meta-algorithmic approaches

The next step involved applying meta-algorithmic approaches to see if any advantages can be obtained through such advanced consensus approaches. A meta-algorithm is a higher-level algorithm that combines more fundamental algorithms to obtain results that are as good or better than the original basis algorithms. For this research, we looked to (Simske, 2013) for descriptions of a library of these meta-algorithmic patterns and picked two first-order meta-algorithms: weighted voting and predictive selection.

Figure 7 Comparing GloVe accuracies for validation corpora (V01-V21)

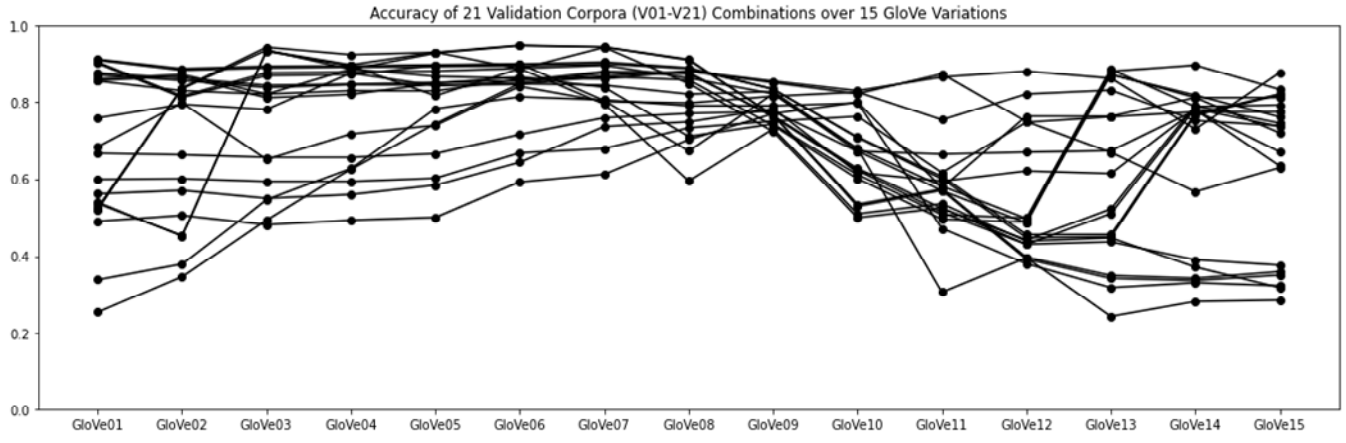


Figure 8 Comparing GloVe variations on each test corpus using V21 validation corpora (see online version for colours)

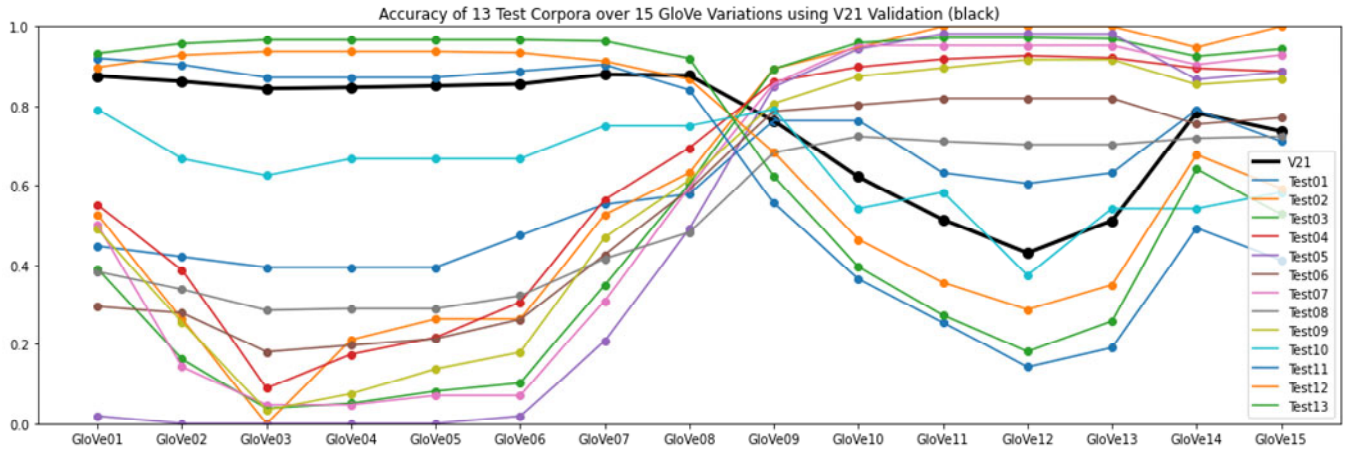


Figure 9 Basic algorithms (blue) compared with GloVe (dotted-black) accuracy across the 13 test corpora for each of the TR01–TR21 training corpora (V01–V21 for GloVe) (see online version for colours)

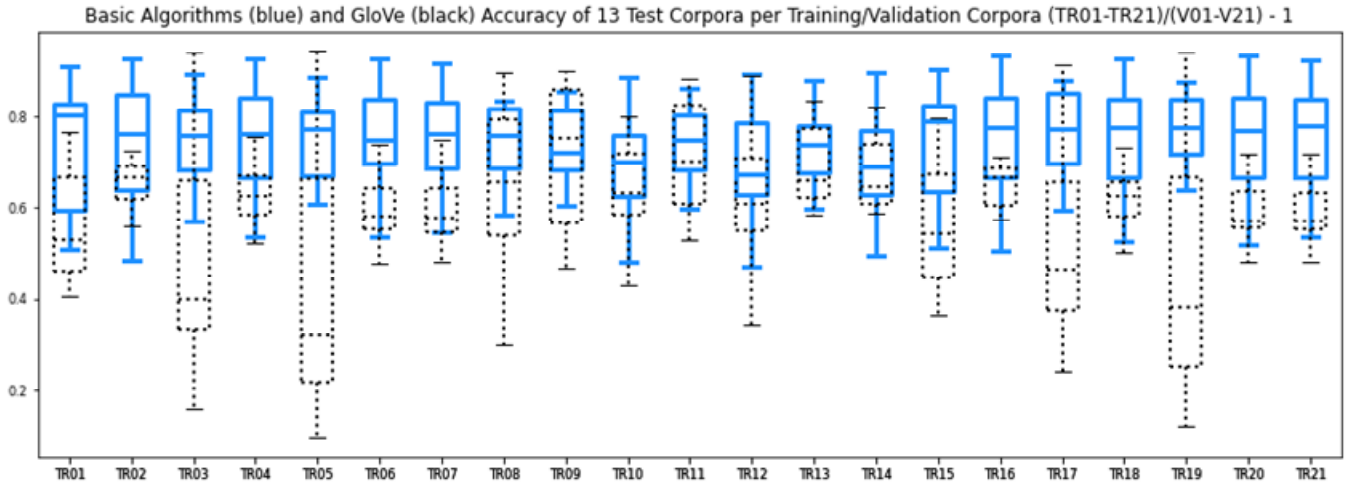
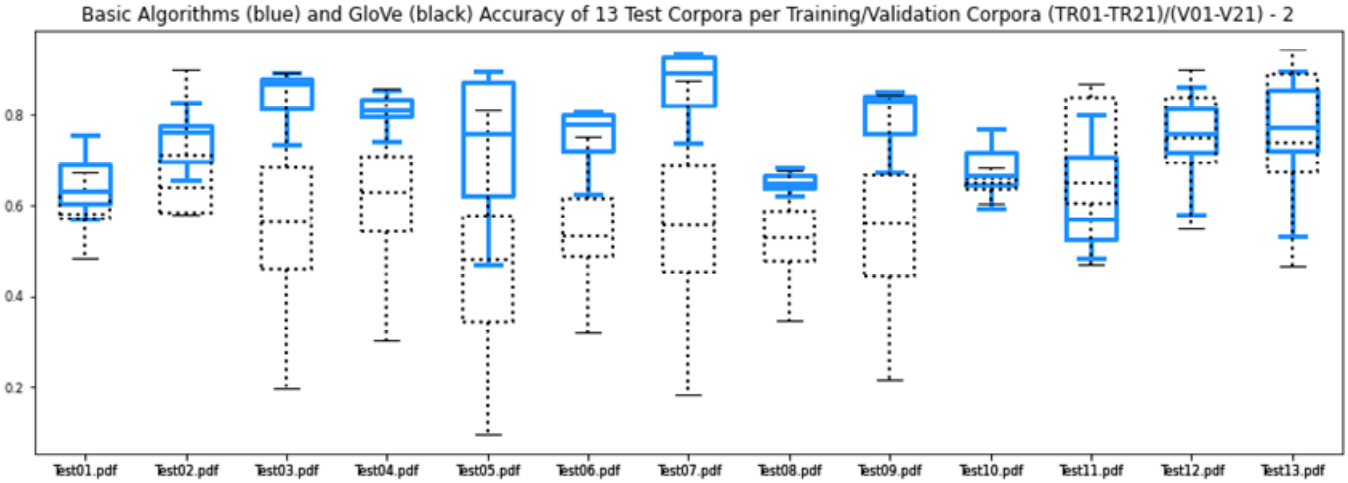


Figure 10 Basic algorithms (blue) compared with GloVe (dotted-black) accuracy across the TR01-TR21 training corpora (V01–V21 validation corpora for GloVe) for each of the 13 test corpora (see online version for colours)



5.2.1 Weighted voting

The weighted voting pattern is often an improvement over the simple voting pattern (Simske, 2013). While the voting pattern weights each component algorithm equally, the weighted voting pattern assigns proportional weights to each of the component algorithms based on their performances in the training corpora. The weighted voting meta-algorithm is depicted in Figure 11.

Figure 11 Weighted voting (see online version for colours)

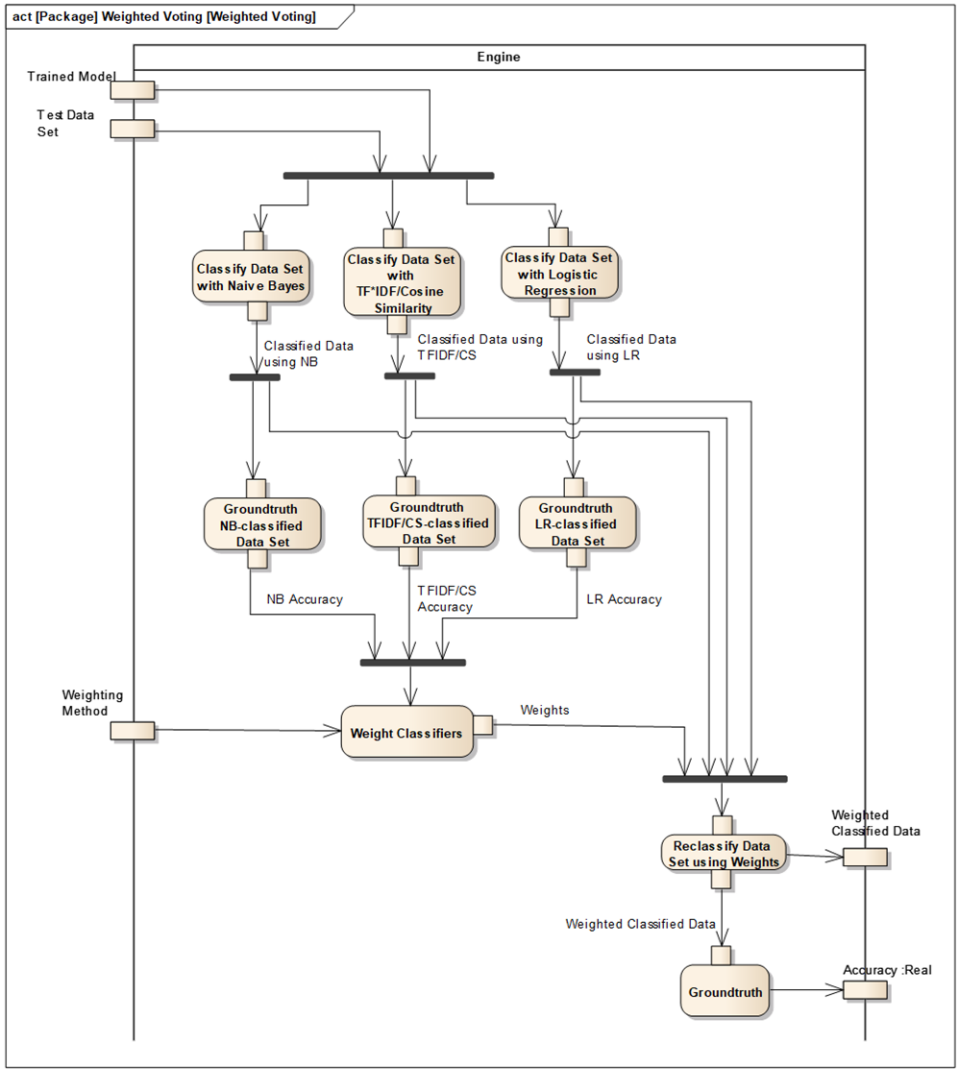


Figure 12 Basic algorithms (blue) compared with weighted voting (dotted-black) accuracy across the 13 test corpora for each of the TR01-TR21 training corpora (see online version for colours)

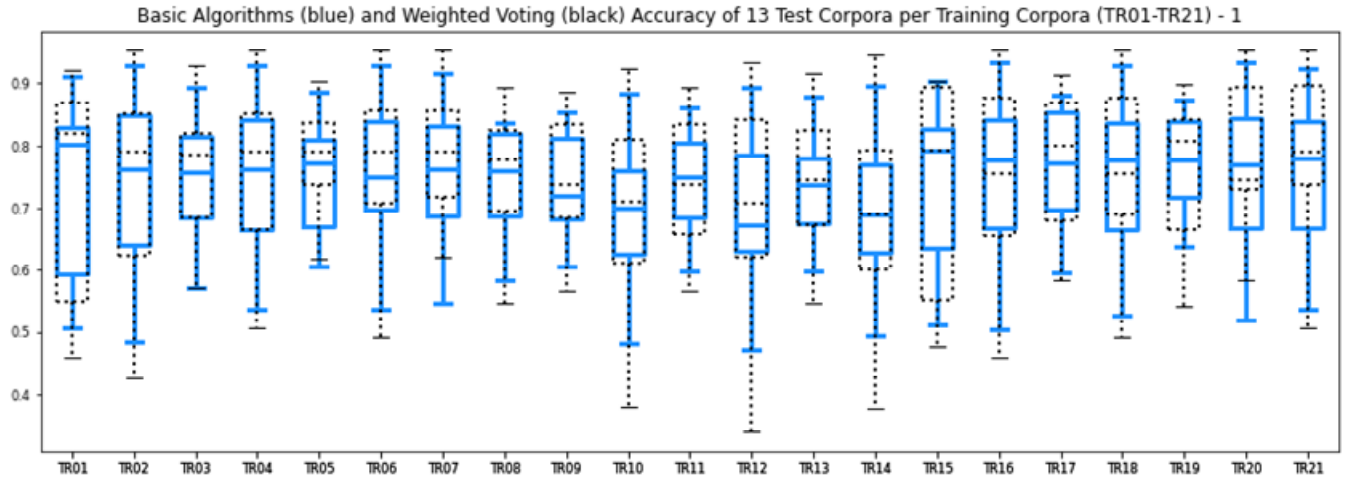


Figure 13 Basic algorithms (blue) compared with weighted voting (dotted-black) accuracy across the 21 training corpora for each of the 13 test corpora (see online version for colours)

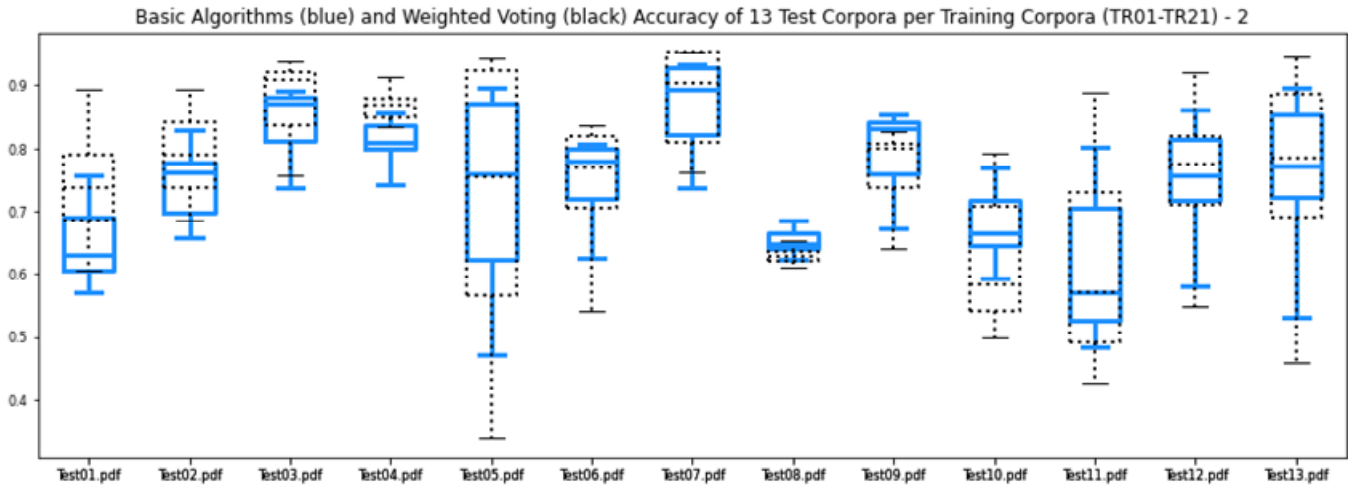
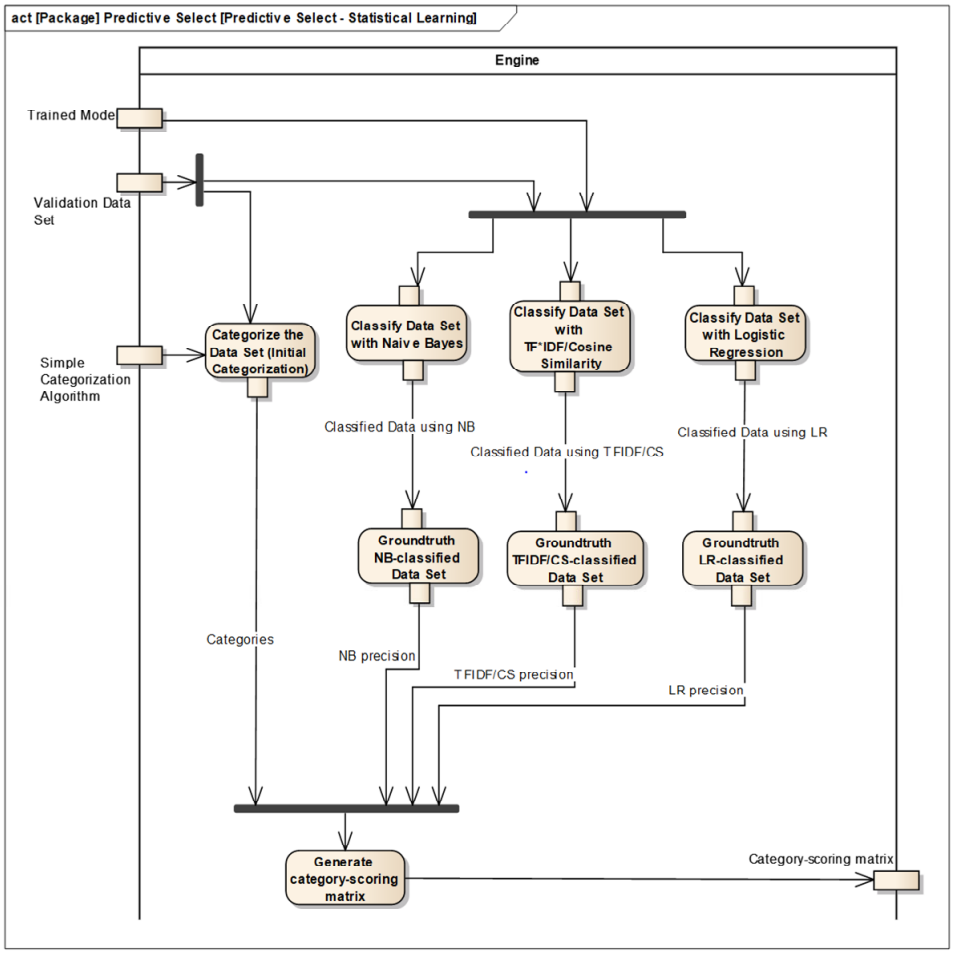


Figure 14 Predictive selection meta-algorithm: statistical learning phase (see online version for colours)



Weighting performed using five methods: *accuracy*, $\frac{1}{error}$, *accuracy*^{2}, $\frac{1}{\sqrt{error}}$, and an information-theory based optimal approach (Lin et al., 2003) to weighting of the form in equations (1) and (2).

$$W_j = \ln\left(\frac{1}{N_{classes}}\right) + \ln\left(\frac{p_j}{e_j}\right), \tag{1}$$

where

$$e_j = \frac{1 - p_j}{N_{classifiers} - 1}. \tag{2}$$

In all cases, we obtained little variation in the results among the five weighting methods, though they did provide mostly improved results when compared to the component algorithms taken together (i.e., non-weighed voting). In some, weighted voting performed worse than the best of the three component algorithms. This can be attributed to the small differences in performance among Naïve Bayes, TF*IDF/cosine similarity, and LR by themselves (Figure 5). The weighted voting algorithm, in many cases, the two relatively inferior component algorithms agree on the classification and therefore overpower the superior component algorithm, which was consistently Naïve Bayes. The results comparing the two approaches are summarised in Figures 12 and 13.

5.2.2 Predictive selection

Predictive selection, like weighted voting, is a first order meta-algorithm (Simske, 2013). For this method, a separate preliminary categoriser is introduced to bin the input dataset and individual component classifiers are chosen to operate on each of those bins. The idea is to select a single component algorithm for each category that provides the best precision for the category. Predictive selection is comprised of two phases: the statistical learning phase (Figure 14) and the run-time phase (Figure 15).

For the first phase, we tried 16 preliminary categorisers. The first involved using a simple pattern match algorithm (described above) on the training corpora and the rest was using the 15 GloVe variations already calculated (Figure 7). The statistical learning phase for each trial provided us with the data to generate the category-scoring matrices.

The second phase of predictive selection, the run-time phase, uses the learnt best algorithms from the training phase (i.e., using the category-scoring matrices). Categorisation during the run-time phase is then performed on the test corpora the same way as the training/validation steps during the statistical learning phase. For our runs, Naïve Bayes was the overwhelming choice regardless of the initial categorisation, followed by LR.

The results of both phases are depicted in Figures 16 and 17 and shows tremendous variation across test corpora. However, in general, predictive selection performed better than weighted voting (and non-weighted voting). In some unusual cases, though we saw a slight degradation of performance, we observed generally consistent results regardless of preliminary categoriser used. For example, using the V21 validation corpora which show a dramatic dip beyond using eight anchor points, Figure 18 shows consistently flat accuracy results given any test corpus (Test01-Test13).

Figure 15 Predictive selection meta-algorithm: run-time phase (see online version for colours)

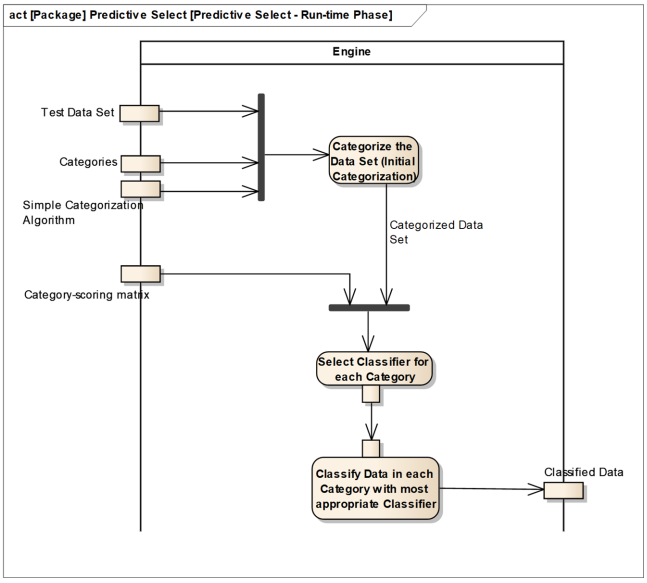


Figure 16 Basic algorithms (blue) compared with predictive selection (dotted-black) accuracy across the 13 test corpora for each of the TR01-TR21 training corpora (see online version for colours)

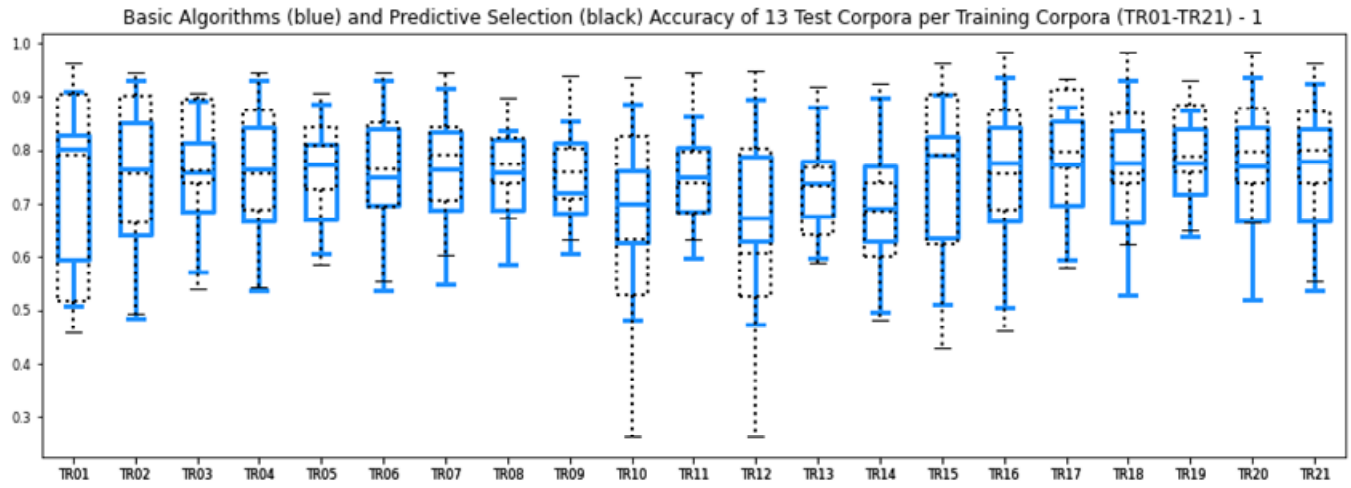


Figure 17 Basic algorithms (blue) compared with predictive selection (dotted-black) accuracy across the 21 training corpora for each of the 13 test corpora (see online version for colours)

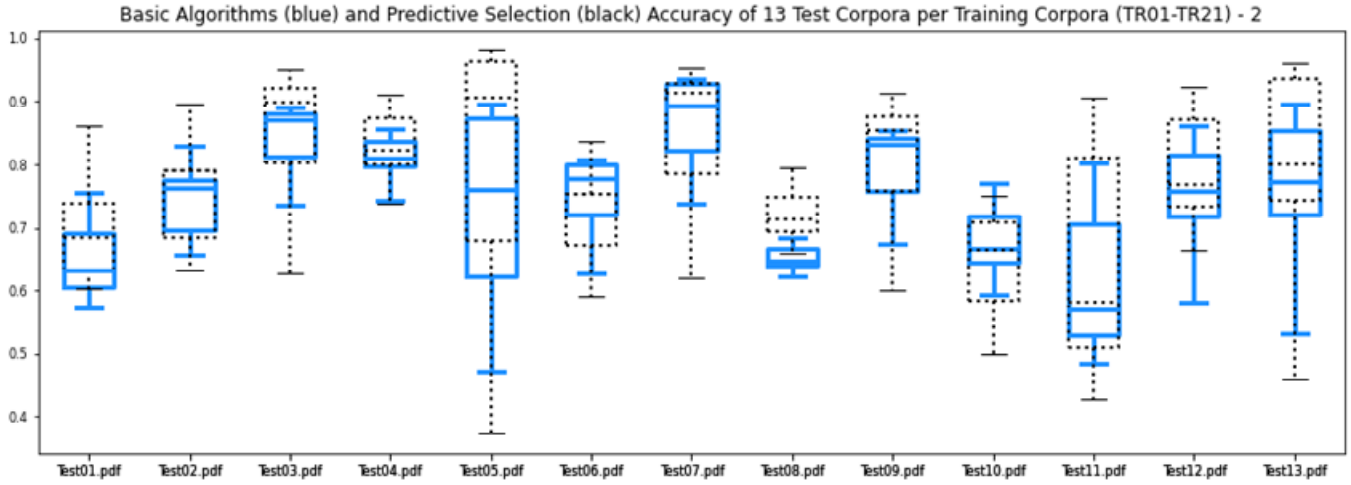
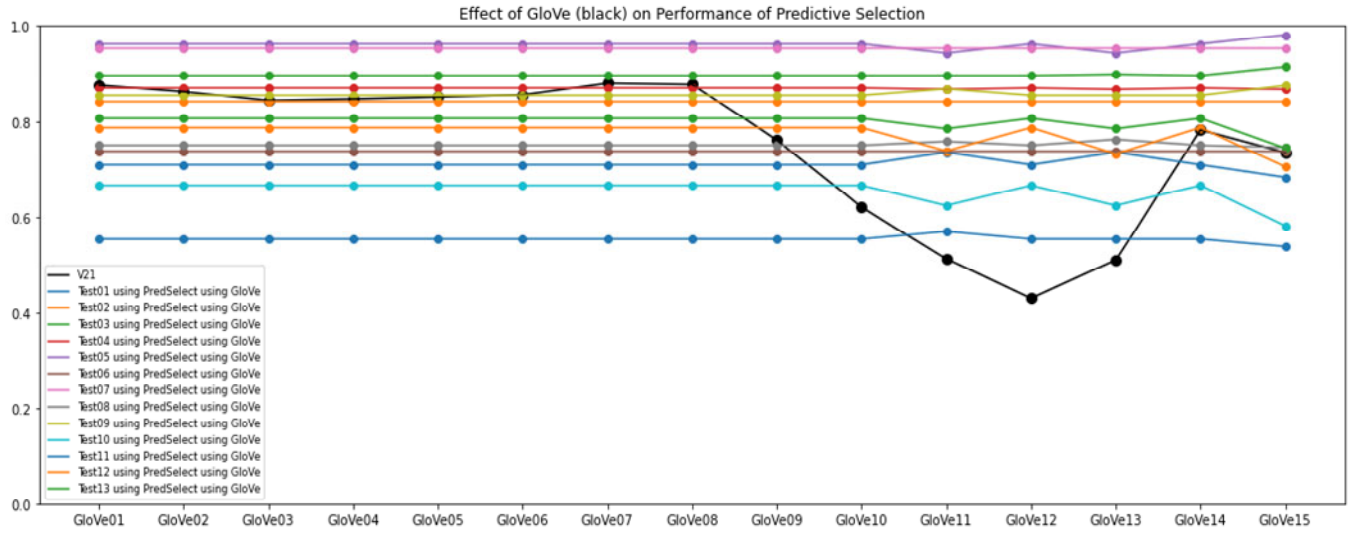


Figure 18 Using the V2 validation corpora, the effect of 15 GloVe variations as a preliminary categoriser on the performance of predictive selection on each of the 13 test corpora remains little-changed (see online version for colours)



This predictive selection performance can be explained by the similar performances exhibited by the three traditional component algorithms from which the predictive selection meta-algorithm chooses: Naïve Bayes, TF*IDF/ cosine similarity, and LR.

5.3 Improvement

To additionally show the advantage of meta-algorithms, we improved the performance of weighted voting by adding a fourth component algorithm. As an illustrative example, we took the worst-performing set of traditional algorithms (TR07, Figure 5) and added GloVe07 to get an improvement in the classification of Test13 from 0.74 to 0.88. The results for Test13 are summarised in Table 5. Note that this improvement is much more pronounced in Test13 because of the poor results from the traditional algorithms for this Test corpus. In general, improvements are not universal by adding GloVe as a fourth component algorithm since our GloVe implementations were highly inconsistent. A mean improvement of 0.03 (from 0.75 to 0.77) has been observed as the poor-performing GloVe variants performed worse than the traditional algorithms and weighed-down the improvements.

Table 5 Accuracies of using various algorithms and the effect of GloVe on weighted voting for Test13 using TR07 only.

Algorithm or meta-algorithm		*Accuracy*
1	Naïve Bayes	0.80
2	TF*IDF/cosine similarity	0.79
3	Logistic regression	0.64
4	GloVe07	0.95
Weighted voting (all variations) using 1, 2 and 3		0.74
Weighted voting (all variations) using 1, 2, 3 and 4		0.88

With predictive selection, a potential improvement could be had by picking a much better preliminary categoriser than a

simple pattern match or the 15 GloVe variations, but for this study, none could be identified.

5.4 Limitations

We have identified risks to validity due to several limitations:

- *Limited publicly available ground-truthed data:* While there is some body of work that currently exist for stakeholder requirements classification, few publicly-available ground-truthed requirements corpora exist. Perhaps the most referenced is the OpenScience tera-PROMISE repository (Cleland-Huang, 2007) of 625 labelled FRs and NFRs, with the NFRs broken down into a set of quality attribute requirements. Other work, such as Sainani et al. (2020) involve private data processed by multiple subject-matter experts over several weeks. For our purposes, every data point used for this study involved initial ground-truthing which took many hours of review. While we eventually ended up with five training corpora with over 1 million words and 13 test corpora with over 2,000 requirements. For GloVe, we settled on a generic model trained with Wikipedia and newswire text.
- *Limited variety:* One of the effects of limited ground-truthed data is we were limited to the four industries to which we had close ties – defence, energy, transportation, and construction, and heavily weighted to defence contracts. With the variety of system types, we expect that with representations from other industries, we would have obtained better results, particularly with classifying system requirements, as industry-specific systems have oftentimes industry-specific terms and acronyms.
- *Parsing imperfections:* Parsing of corpora involved ingesting PDF files and was rudimentary, dependent on properly written requirements with *shall* statements ending in periods. Statements that included a *shall* but had multiple bullet points were processed only based on the first period. For example, a requirement of the form on Table 6 translates to a single requirement '*the system shall: statement 1*'. and the rest of the requirements, statement 2 through statement *n*, are ignored.
- *Poorly written requirements:* While it is not expected that requirements follow all of INCOSE's (2017) recommendations for writing requirements, a certain level of quality was expected. With the set of corpora at our disposal, the larger projects appeared to have better-written SOWs, presumably because their potentially larger cost and schedule risk necessitated more experienced systems engineers to write the SOWs.
- *Disguised requirements:* This study looked at classifying requirements into governance and system, the former type levied on the contractor and the latter levied on the system. While most requirements have clear-cut classifications, a few straddle the line. In particular, some requirements initially appear to be governance requirements but are really system requirements. For example, "the contractor shall design the module for reliability" is a requirement levied on the contractor/designer but the implications are on the system being developed. We noticed that a construction SOW we had (Test08) was rife with these requirements that are levied on the design-build company but describe design constraints of the project. Other requirements, such as those focusing on cost, are even more blurred, as cost is a responsibility of both project manager and engineer. One approach to alleviate this problem is multi-label classification similar to that espoused in Sabir et al. (2020) and using a rating system instead of a binary classification.

Table 6 Multiple requirements written as bullets

The system shall
• Statement 1.
• Statement 2.
• …
• Statement *n*.

6 Conclusions

We initially ran 40 algorithms and meta-algorithm variations trained over 21 training corpora combinations tested over 13 test corpora for a total of 10,920 combinations. The results are summarised as follows and on Table 7.

6.1 Training and validation corpora

In the absence of publicly-available ground-truthed training corpora, a substitution using the following was useful and provided reasonable results:

1 several medium to large SOWs and specifications documents and combinations of them

2 Wikipedia + Gigaword combination for GloVe.

6.2 Comparison of algorithms and meta-algorithms

A summary of the all the results is shown in Table 7. Figure 3 is a slice of that summary using TR21/V21 only, and Table 5 shows a different slice. These slices provide more pronounced differences in the algorithmic and meta-algorithmic implementations. Nonetheless, Table 7, regardless of the sub-optimal variations of the component algorithms, shows an advantage of meta-algorithmic approaches.

The simple pattern match, as expected, did not do so well and results were extremely sensitive to a subject matter expert picking the right words for matching.

Table 7 Algorithm/meta-algorithm performance summary across all test corpora

	Algorithm	*Type*	*Training*	*Validation*	*Test*	*Results across all test corpora*
1	Simple pattern match	Exact match	N/A	N/A	Table 3	0.69
2	Naïve Bayes	ML algorithm	Table 2	N/A	Table 3	0.76
3	TF*IDF/cosine similarity	ML algorithm	Table 2	N/A	Table 3	0.76
4	Logistic regression	ML algorithm	Table 2	N/A	Table 3	0.75
5	GloVe09 (best variant of Glove01-GloVe15)	ML algorithm (deep learning) (Pennington et al., 2014)	Pennington et al. (2014) and Parker et al. (2011)	N/A	Table 3	Best (GloVe09): 0.74 All (GloVe01-15): 0.61 Highly inconsistent per variant
6	Weighted voting using 2, 3, and 4	Meta-algorithm with components 2, 3 and 4	Table 2	N/A	Table 3	0.75 Only as good as 2, 3 and 4 will allow
7	Predictive selection using 1 as prelim. categoriser	Meta-algorithm with components 2, 3 and 4	Table 2	N/A	Table 3	0.75 Only as good as 2, 3 and 4 will allow
8	Predictive selection using 5 variants as prelim. categoriser	Meta-algorithm with components 2, 3 and 4	Pennington et al. (2014) and Parker et al. (2011)	Table 2	Table 3	0.76 Highly consistent for a particular test corpus regardless of GloVe performance as a preliminary categoriser. Dependent on quality of 2, 3 and 4.
9	Weighted voting using 2, 3, 4 and 5	Meta-algorithm with components 2, 3, 4 and 5	Table 2	N/A	Table 3	0.77 Only as good as 2, 3, 4 and 5 will allow

Naïve Bayes, despite its simplicity, provides the best results compared with processing time. This is consistent with what has been documented in the past (Zhang, 2004; Kupervasser, 2014).

TF*IDF/cosine similarity and Naïve Bayes, despite using the same frequency vectors, varied in the training corpora, but performed similarly on the test corpora. Naïve Bayes, however, generally outperformed the former.

GloVe was heavily influenced by the number of anchor words, and depending on the validation corpora, produced highly sensitive results (Figure 7) but did notice that fewer anchor words (GloVe01-Glove08) appeared to favour highly governance-heavy corpora, while more anchor words (GloVe09-GloVe15) favoured system-heavy corpora (e.g., see Figure 8). We did see a 'tightness' or convergence on GloVe09 which may indicate that that GloVe variation would be the best to use for future research.

Weighted voting was good but depended a bit on how the inferior algorithms fared. In some cases, the two inferior algorithms overwhelmed the best one and resulted in a misclassification whereas the individual best classifier would have picked the correct one.

Predictive selection, heavily dependent on the component algorithms, improved upon weighted voting even with preliminary categorisers that were not ideal. A lower bound on classification accuracy can be obtained even with these non-ideal preliminary categorisers (Figure 18). GloVe is extremely heavy, and while in many cases, it can be used to improve results, one must keep in mind the resources needed for using GloVe. Perhaps better training and validation corpora would make GloVe the hands-down choice for classification, but for this study, we did not find using GloVe very compelling.

The final results of all these combinations showed how meta-algorithms could be used to stabilise results to improve the lower bound on accuracies given certain basic or traditional component algorithms. In addition, meta-algorithms have the advantage of remaining fresh and relevant no matter what new singular component algorithms are devised in the future. Meta-algorithms can and often get more powerful as these new components are employed. For our study, our GloVe implementation was not compelling by itself because of the enormous variations of results. While GloVe implementations did somewhat improve our meta-algorithmic results, a 60-fold increase in processing time negates a reason for including GloVe as a component unless a superior set of training corpora is identified.

7 Future work/areas of further experimentation and research

The 10,920 combinations of training, validation, test, and algorithms provided some insight on how classifying stakeholder requirements could be performed and improved. Because of a wide variety of parameters that could be substituted for those that we used (such as word anchors for GloVe, the training and validation corpora themselves, various other mixes of component algorithms, other weighting methods, etc.), exhaustive investigation was not

feasible. However, the insights gleaned could be used to drive future similar efforts.

There is a lack of publicly accessible ground truth documents. An effort to provide such documents to the community could be started. We also found that parsing needed some work. Basic parsing of PDF files was performed and provided reasonably good results, but as described in the limitations section above, classification results could benefit from a more robust parsing algorithm.

Classifying requirements not just on words but also on the context headers could prove useful. For example, requirements under the heading 'reliability requirements' could provide additional weight on classifying those requirements as system and not governance.

Realising that classification of SOW requirements is not always black and white, using multi-label classification (Sabir et al., 2020) and providing weights on those could prove useful. Finally, the methods used for this research could be tailored for three or more classifications that are even more fuzzy, such as classifying for quality attributes such as availability, security, sustainability, usability, and others.

References

Abad, Z.S.H. et al. (2017) 'What works better? A study of classifying requirements', *IEEE*, pp.496–501.

Anon. (2018) 'ISO/IEC/IEEE International Standard – systems and software engineering – life cycle processes – requirements engineering', *ISO/IEC/IEEE*, No. 29148:2018(E).

Bird, S., Loper, E. and Klein, E. (2009) *Natural Language Processing with Python*, O'Reilly Media Inc., Sebastapol, CA.

Black's Law Dictionary Free Online Legal Dictionary (n.d.) *What is SHALL?*, 2nd ed. [online] https://thelawdictionary.org/shall/ (accessed 4 February 2022).

Canedo, E.D. and Mendes, B.C. (2020) 'Software requirements classification using machine learning algorithms', *Entropy*, Vol. 22, No. 2, pp.1–20.

Cleland-Huang, J. (2007) *Exploración de base de datos de Atributos de Calidad* [online] http://github.com/Manolomon/tera-promise (accessed 19 November 2021).

Clements, P. et al. (2003) *Documenting Software Architectures: Views and Beyond*, Pearson Education, Inc., Boston.

DAU (n.d.) *Defense Acquisition Guidebook* [online] http://www.dau.edu (accessed 11 August 2021).

Dick, J., Hull, E. and Jackson, K. (2017) *Requirements Engineering*, Springer International Publishing AG, London, UK.

Douglass, B.P. (2016) *Agile Systems Engineering*, Elsevier, Inc., Waltham, MA.

Giannakopoulou, D., Pressburger, T., Mavridou, A. and Schumann, J. (2020) 'Generation of formal requirements from structured natural language', in *Proceedings of the 26th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)*, Pisa, Italy.

Glinz, M. (2020) *CPRE Glossary 2.0* [online] http://www.ireb.org/en/downloads/ (accessed 19 November 2021).

Hefner, R. (2019) *Requirements Management Tutorial*, Caltech, San Diego.

INCOSE (2015) *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, John Wiley & Sons, Inc., San Diego, CA.

INCOSE (2017) *Guide for Writing Requirements*, INCOSE, San Diego, CA.

INCOSE (2022) *Needs, Requirements, Verification, Validation Lifecycle Manual*, INCOSE, San Diego.

Jones, C. (2000) *Software Assessments, Benchmarks, and Best Practices*, Addison Wesley Longman, Inc., New Jersey.

Kossiakoff, A., Biemer, S.M., Seymour, S.J. and Flanigan, D.A. (2020) *Systems Engineering Principles and Practice*, 3rd ed., Wiley, Hoboken, NJ.

Kotonya, G. and Sommerville, I. (1998) *Requirements Engineering: Processes and Techniques*, John Wiley & Sons, Inc., Chichester.

Kupervasser, O. (2014) 'The mysterious optimality of Naïve Bayes: estimation of the probability in the system of 'classifiers'', *Mathematical Theory of Pattern Recognition*, Vol. 24, No. 1, pp.1–10.

Ladyman, J. and Wiesner, K. (2020) *What is a Complex System?*, Yale University Press, New Haven, CT.

Lane, H., Howard, C. and Hapke, H.M. (2019) *Natural Language Processing in Action*, Manning Publications Co., Shelter Island, NY.

Lin, X., Yacoub, S., Burns, J. and Simske, S. (2003) 'Performance analysis of pattern classifier combination by plurality voting', *Pattern Recognition Letters*, Vol. 24, No. 12, pp.1959–1969.

Lucio, L., Rahman, S., Cheng, C-H. and Mavin, A. (2017) *Just Formal Enough? Automated Analysis of EARS Requirements*, Moffet Field, CA.

Mahmoud, A. and Williams, G. (2016) 'Detecting, classifying, and tracing non-functional software requirements', *Requirements Engineering*, Vol. 21, No. 3, pp.357–381.

Mavin, A., Wilkinson, P., Harwood, A. and Novak, M. (2009) *EARS (Easy Approach to Requirements Syntax)*. Atlanta, Georgia.

NASA (n.d.) *Appendix C: How to Write a Good Requirement* [online] http://www.nasa.gov/seh/appendix-c-how-to-write-a-good-requirement (accessed 11 November 2021).

Parker, R. et al. (2011) *English Gigaword*, 5th ed. [online] http://catalog.ldc.upenn.edu/LDC2011T07 (accessed 15 September 2021).

Pedregosa, F. et al. (2011) 'Scikit-Learn: machine learning in Python', *Journal of Machine Learning Research*, Vol. 12, pp.2825–2830.

Pennington, J., Socher, R. and Manning, C.D. (2014) *GloVe: Global Vectors for Word Representation* [online] http://nlp.stanford.edu/projects/glove (accessed 15 September 2021).

Rozanski, N. and Woods, E. (2011) *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*, 2nd ed., Addison-Wesley Professional, Upper Saddle River, NJ.

Sabir, M., Chrysoulas, C. and Banissi, E. (2020) 'Multi-label classifier to deal with misclassification in non-functional requirements', *International Journal of Information Technology*, Vol. 12, No. 1, pp.101–110

Sainani, A., Anush, P.R., Joshi, V. and Ghaisas, S. (2020) 'Extracting and classifying requirements from software engineering contracts', *IEEE*, pp.147–157.

Simske, S.J. (2013) *Meta-Algorithmics: Patterns for Robust, Low Cost, High Quality Systems*, John Wiley & Sons, Ltd., New York, NY.

Zhang, H. (2004) *The Optimality of Naïve Bayes*, Miami Beach, Florida.