A Novel Approach for the Detection of Web Service Anti-Patterns Using Word Embedding Techniques

Sahithi Tummalapalli[1], Lov Kumar[1(✉)], Lalitha Bhanu Murthy Neti[1], Vipul Kocher[2], and Srinivas Padmanabhuni[2]

[1] BITS Pilani Hyderabad, Hyderabad, India
{P20170433,lovkumar,bhanu}@hyderabad.bitspilani.ac.in
[2] Testaing.Com, Bengaluru, India
{vipulkocher,srinivas}@testAing.com

Abstract. An anti-pattern is defined as a standard but ineffective solution to solve a problem. Anti-patterns in software design make it hard for software maintenance and development by making source code very complicated for understanding. Various studies revealed that the presence of anti-patterns in web services leads to maintenance and evolution-related problems. Identification of anti-patterns at the design level helps in reducing efforts, resources, and costs. This makes the identification of anti-patterns an exciting issue for researchers. This work introduces a novel approach for detecting anti-patterns using text metrics extracted from the Web Service Description Language (WSDL) file. The framework used in this paper builds on the presumption that text metrics extracted at the web service level have been considered as a predictor for anti-patterns. This paper empirically investigates the effectiveness of three feature selection techniques and the original features, three data sampling techniques, the original data, four word embedding techniques, and nine classifier techniques in detecting web service anti-patterns. Data Sampling techniques are employed to counter the class imbalance problem suffered by the data set. The results confirm the predictive ability of text metrics obtained by different word embedding techniques in predicting anti-patterns.

Keywords: Web service · Word embedding techniques · Machine learning · Classifier techniques · Class imbalance · Anti-pattern · Text metrics

1 Introduction

Service-Oriented Architecture (SOA) is defined as: "a style of multi-tier computing that helps organizations share logic and data among multiple applications and usage modes [6]". *SOA* is an architecture used to create systems based

O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12955, pp. 217–230, 2021.
https://doi.org/10.1007/978-3-030-87007-2_16

on autonomous coarse-grained and loosely coupled interactions between components called services. Each service bears behavior and processes through contracts, including messages sent to recognizable addresses (called endpoints). The software industry has considered SOA as a structure to improve the ordering between the IT industry and the business models to adapt more flexibly to the businesses' ever-changing requirements, increasing the business agility. The ever-changing requirement of the business makes many modules introduced into the initial software design, evolving it over time and making it very complicated to maintain and understand. The evolution of software design to fit user and business demands degrades the software's quality. It leads to an ineffective solution to a frequently occurring problem, called *Anti-Pattern*. In this work, We considered the following four web service anti-patterns, namely: AP1: GOWS(God Object Web Service), AP2:FGWS(Fine-Grained Web Service), AWS: Ambiguous Web Service, AP4: CWS(Chatty Web Service). Various studies revealed that the presence of anti-patterns in web services leads to maintenance and evolution-related problems. Identification of anti-patterns at the design level helps in reducing efforts, resources, and costs. This makes the identification of anti-patterns an interesting problem for the researchers.

Segev et al. [7] in their work analyzed two methods, namely: Term Frequency/Inverse Document Frequency (Tf-IDF) and context analysis for processing text. The authors have explored WSDL files and free textual descriptors publicly available in service repositories for analyzing the services. By analyzing the WSDL and free text descriptions, authors proved by their approach that web service usage can be broadened by exploiting the data present in the web for building rich context for client queries rather than burdening users to marginalize their services with formal concepts and explanations. This work motivated us to utilize the TF/IDF approach and three other word embedding techniques to generate text metrics from the WSDL description files, which are further used as input for building autonomous models to detect anti-patterns in web services. In our previous work [8–10], We have build web service anti-pattern detection techniques using the object-oriented metrics and the WSDL metrics that are generated from the web service description language (WSDL) files as input. This paper empirically investigates the effectiveness of three feature selection techniques and the original features, three data sampling techniques, the original data, four word embedding techniques, and nine classifier techniques in detecting web service anti-patterns. Data Sampling techniques are employed to counter the class imbalance problem suffered by the data set.

2 Literature Survey

Pietrzak and Walter [5] defined and analyzed various relationships among the smells and provided hints on how they could be exploited to reduce anti-patterns detection. Authors performed experiments in the Jakarta Tomcat code to prove that knowledge about identified smells improves detecting other smells existing in the code. Jaafar et al. [3] argued in his paper that classes taking part in anti-pattern and patterns of software designs have dependencies with other classes,

i.e., unvarying and mutating dependencies, that may spread issues to different classes. In this paper, the authors have empirically investigated the consequences of dependencies in the object-oriented system by focusing and analyzing the relationship between co-change and static dependencies and change proness, fault proness, and fault types classes are exhibiting. The hypothesis is validated on six design anti-patterns and ten anti-patterns in 39 releases of XercesJ, JFreeChart, and ArgoUML. Experimental results showed that the classes with anti-patterns are more prone to fault proness when compared to other classes. The limitations of this work are 1. Authors have conducted their experiments on a limited set of anti-patterns and design patterns. 2. The metrics used in the study are also limited. Velioglu and Selcuk [11] developed a Y-CSD tool that detects and reduces anti-patterns and code smells in the software project. The proposed tool is used to detect two anti-patterns, namely Brain Method and Data Class. Y-CSD uses structural analysis for the detection of code smells and anti-patterns. Kumar and Sureka [4] proposed an approach for the automatic detection of anti-patterns by static analysis of the source code. In this paper, the author proposed that the aggregate values of the source code metrics computed at the web-service level can be used as predictors for anti-pattern detection. In this paper, the author has empirically investigated the application of eight machine learning algorithms, i.e., Bagging, Multilayer Perceptron, Random Forest, Naive Bayes, Decision Tree, Logistic Boost, AdaBoost, Logistic regression, four data sampling techniques, namely: Downsampling, Random Sampling, and Synthetic Minority oversampling Technique (SMOTE) and four feature selection techniques, i.e., Information Gain, Symmetric Uncertainty, Gain Ratio and OneR in the prediction of anti-patterns. Borovits et al. [2] proposed technique for the detection of linguistic anti-patterns in infrastructure as code (Iac) scripts. The authors employed deep learning and word embedding techniques for the proposed approach. Further, the authors build the abstract syntax tree of Iac code units to create their code embedments. Authors validated their approach on the dataset extracted from open source repositories, and experimental results showed an accuracy ranging from 0.78–0.91 in detecting linguistic anti-patterns in Iac.

3 Proposed Framework and Research Background

In this work, experiments were carried on a downloaded dataset from GitHub repository[1]. Figure 1 shows the quantity of web services in which each of the anti-patterns exist (%AP) and does not exist (%NAP) in the dataset.

	GOWS	FGWS	DWS	CWS
# AP	21	13	14	21
% AP	9.29	5.75	6.19	9.29

Fig. 1. Percentages(%AP) and number(#AP) of anti-patterns in dataset

[1] https://github.com/ouniali/WSantipatterns.

Figure 2 shows the proposed framework for the experimental work. The experimental dataset has WSDL description files collected from various domains like weather, education, finance, etc. The first step comprises generating text metrics from the WSDL file by applying four word embedding techniques, namely: Term frequency-inverse Document Frequency (Tf-IDF), Continuous Bag Of Words (CBOW), Global Vectors for Word Representation (GloVe), and SKip Gram (SKG). Each of these techniques is producing around 450–1200 features for each of the WSDL files. There is a chance that some of these features are irrelevant in the detection of anti-patterns. To remove such features, we used three feature selection techniques to select the significant features in the next step. As we have discussed in Sect. 3, the dataset considered for the experiment is suffering from the class imbalance problem. Therefore, we used three data sampling techniques, namely: Synthetic Minority Oversampling Technique (SMOTE), BSMOTE (Borderline SMOTE), and ADASYN (Adaptive Synthetic), to overcome the class imbalance problem. Further, We used nine different classifier techniques to generate models for the prediction of web service anti-patterns. Finally, we did the comparative analysis of various models developed to predict web service anti-patterns using performance measures such as Accuracy, Area under Curve (AUC), and F-measure.

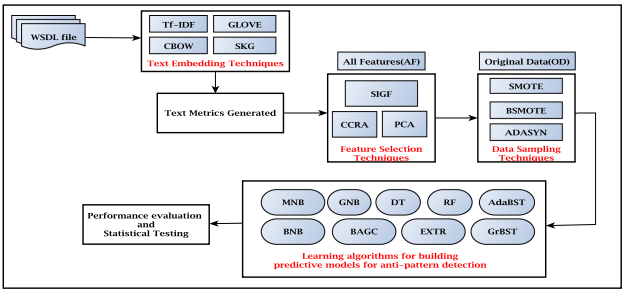

Fig. 2. Proposed framework

3.1 Word Embedding Techniques for the Generation of Text Metrics

In this work, we use four different word embedding techniques to generate text metrics that are later used as input for generating the models for detecting web service anti-patterns. As discussed in Sect. 3, the considered dataset has a collection of WSDL description files. All the word embedding techniques are applied on each of the WSDL files in which each line of code is considered text to generate the text metrics. A brief of each of the word embedding technique used in this work are given below:

– ***Term frequency- Inverse Document Frequency (Tf-IDF):*** is a numerical approach that shows the importance of a word in a file. Tf-IDF value

increases in proportion with the number of times a word surfaces in a file and is balanced by the number of files in the corpus that has the word.

$$Tf(w) = \frac{\text{(no. of times w surfaces in a file)}}{\text{(total no. of words in the file)}}$$
$$IDF(w) = \log_e\left(\frac{\text{total no. of files}}{\text{no.of files with w in it}}\right) \quad (1)$$

- ***Continuous Bag Of Words(CBOW):*** CBOW model predicts the center words based on the context of the surrounding words. CBOW predicts the center word based on the context_window words.
- ***SKip Gram(SKG):*** Skip-gram model computes the probability of the center word appearing with the context words by computing the similarity with the dot product, then it converts the similarity into probability by passing it through the soft-max functions.
- ***Global Vectors for Word Representation(GLOVE):*** Glove uses matrix factorization techniques on the word-context matrix. In this technique, we construct a matrix of co-occurrence information, in which we count each word shown in rows and the frequency of occurrence of the word in a particular context shown in the column. For each word, we search for the context terms within a specified range defined by the window size before the word and a window size post the word.

3.2 Feature Selection Techniques

As discussed in Subsect. 3.1, we applied four different word embedding techniques individually on each of the WSDL description files for generating text metrics as features. Each of the techniques produced around 450–1200 features for every WSDL file. All the features generated may not be significant in the detection of anti-patterns. Hence, it is vital to remove all the irrelevant and insignificant features from the data before generating the models. Firstly, we applied a set of feature selection and ranking techniques used in previous work [9] to select the significant features (SIGF). Then we applied two other techniques, Compactness Centroid based Average(CC_ra) [1], Principal Component Analysis(PCA) for selecting the relevant and significant features. We use the features selected by different feature selection techniques and the original features as input for building the models to detect web service anti-patterns.

3.3 Data Sampling Techniques

As discussed in Sect. 3.1, the dataset considered for experiments is suffering from the class imbalance problem. To counter this problem, We used three different sampling techniques in this work, namely: Synthetic Minority Oversampling Technique (SMOTE), BSMOTE (Borderline SMOTE), and ADASYN (Adaptive Synthetic). We developed the models to detect anti-patterns using the data after applying different sampling techniques and the original data.

3.4 Classifier Techniques

We apply various classifiers such as Naive Bayes classifier with different kernels, i.e., Multinomial (MNB), Bernoulli (BNB) and Gaussian (GNB), Decision Tree (DT), Bagging classifier (BAGC), Random Forest (RF), Extra Randomized Tree classifier (EXTR), AdaBoost (AdaBST) and Gradient Boost (GRBST) for training the models for the detection of various anti-patterns considered in this study. Furthermore, while training the models, we split the data in the ratio of 80:20 for training and testing.

4 Experimental Results

This paper empirically investigates the effectiveness of three feature selection techniques and the original features, three data sampling techniques, the original data, four word embedding techniques, and nine classifier techniques in detecting web service anti-patterns. The total number of predictive models built to detect web service anti-patterns using text metrics as input are $4 \times 4 \times 4 \times 9 \times 4 = 2304$. Performance metrics such as accuracy, Area Under Curve (AUC) are used for evaluating the predictive ability of the models generated for the detection of web service anti-patterns. Table 1 shows the accuracy values for all the models generated to detect GOWS anti-pattern. From Table 1, we observed that the:

- We observed that the models trained on data after applying sampling techniques show good performance compared to the models trained on the original data.
- It is observed that the model developed by data after applying the data sampling technique SMOTE shows better performance.
- We observed that the anti-pattern prediction model developed using the features generated by applying the word embedding technique Tf-IDF shows the best performance with the mean accuracy value of 91.93.
- It is also observed that the mean accuracy value of the model developed using the features selected as significant features(SIGF) as input is higher than the models developed using the features selected by CC_ra, and PCA as input.
- We observed that the model trained using ensemble technique Extra Trees classifier with a mean accuracy of 95.35 is higher when compared to the models trained using other classifier techniques.

5 Comparative Analysis

This section compares the models' performance using various word embedding techniques, data sampling techniques, feature selection techniques, and classifier techniques to detect multiple anti-patterns using box-plots, descriptive statistics, and statistical hypothesis testing.

Table 1. AUC Value of GOWS anti-pattern

Feature selection	Data sampling technique	Word embedding technique	MNB	BNB	GNB	DT	BAGC	RF	EXTR	AdaBST	GrBST
AF	**OD**	**tf-idf**	0.55	0.91	0.89	0.89	0.91	0.91	0.91	0.90	0.86
AF	**OD**	**cbow**	0.62	0.87	0.66	0.88	0.91	0.90	0.91	0.85	0.88
AF	**OD**	**skg**	0.62	0.86	0.64	0.88	0.91	0.91	0.93	0.90	0.86
AF	**OD**	**Glove**	0.76	0.84	0.65	0.84	0.92	0.92	0.91	0.88	0.84
AF	**SMOTE**	**tf-idf**	0.98	0.99	0.98	0.92	0.95	0.97	0.99	0.95	0.90
AF	**SMOTE**	**cbow**	0.77	0.53	0.82	0.92	0.81	0.90	0.92	0.90	0.88
AF	**SMOTE**	**skg**	0.76	0.52	0.81	0.90	0.83	0.94	0.95	0.93	0.93
AF	**SMOTE**	**Glove**	0.77	0.52	0.80	0.91	0.81	0.93	0.96	0.93	0.90
AF	**BSMOTE**	**tf-idf**	0.98	0.99	0.97	0.94	0.96	0.98	0.99	0.93	0.91
AF	**BSMOTE**	**cbow**	0.79	0.53	0.85	0.95	0.84	0.96	0.94	0.93	0.93
AF	**BSMOTE**	**skg**	0.80	0.52	0.84	0.94	0.86	0.96	0.96	0.94	0.95
AF	**BSMOTE**	**Glove**	0.80	0.52	0.83	0.88	0.83	0.96	0.96	0.92	0.93
AF	**ADASYN**	**tf-idf**	0.98	1.00	0.98	0.92	0.92	0.99	0.99	0.95	0.86
AF	**ADASYN**	**cbow**	0.73	0.53	0.76	0.82	0.74	0.85	0.85	0.86	0.86
AF	**ADASYN**	**skg**	0.71	0.53	0.78	0.88	0.79	0.88	0.87	0.87	0.87
AF	**ADASYN**	**Glove**	0.72	0.52	0.74	0.83	0.73	0.89	0.93	0.82	0.82
SIGF	**OD**	**tf-idf**	0.93	0.99	0.72	0.92	0.91	0.91	0.91	0.91	0.89
SIGF	**OD**	**cbow**	0.62	0.87	0.64	0.89	0.91	0.90	0.89	0.90	0.88
SIGF	**OD**	**skg**	0.63	0.86	0.62	0.87	0.91	0.91	0.92	0.89	0.85
SIGF	**OD**	**Glove**	0.77	0.85	0.65	0.87	0.92	0.92	0.91	0.86	0.82
SIGF	**SMOTE**	**tf-idf**	0.99	0.99	0.96	0.92	0.96	0.97	0.99	0.96	0.91
SIGF	**SMOTE**	**cbow**	0.77	0.52	0.82	0.92	0.82	0.94	0.93	0.90	0.88
SIGF	**SMOTE**	**skg**	0.76	0.52	0.81	0.90	0.84	0.94	0.95	0.92	0.91
SIGF	**SMOTE**	**Glove**	0.77	0.52	0.80	0.91	0.82	0.93	0.94	0.89	0.92
SIGF	**BSMOTE**	**tf-idf**	0.99	0.99	0.96	0.93	0.98	0.97	0.98	0.92	0.92
SIGF	**BSMOTE**	**cbow**	0.79	0.52	0.85	0.96	0.84	0.96	0.96	0.93	0.93
SIGF	**BSMOTE**	**skg**	0.80	0.52	0.84	0.94	0.86	0.96	0.96	0.94	0.94
SIGF	**BSMOTE**	**Glove**	0.80	0.52	0.83	0.90	0.84	0.96	0.96	0.91	0.90
SIGF	**ADASYN**	**tf-idf**	0.99	0.99	0.96	0.91	0.89	0.97	0.99	0.90	0.87
SIGF	**ADASYN**	**cbow**	0.73	0.53	0.75	0.84	0.76	0.83	0.85	0.86	0.84
SIGF	**ADASYN**	**skg**	0.71	0.53	0.78	0.89	0.80	0.87	0.89	0.87	0.87
SIGF	**ADASYN**	**Glove**	0.72	0.52	0.75	0.84	0.75	0.91	0.91	0.84	0.74
CCRA	**OD**	**tf-idf**	0.95	0.97	0.90	0.90	0.91	0.91	0.91	0.92	0.92
CCRA	**OD**	**cbow**	0.91	0.89	0.65	0.87	0.91	0.89	0.91	0.88	0.86
CCRA	**OD**	**skg**	0.91	0.88	0.63	0.88	0.91	0.90	0.92	0.90	0.89
CCRA	**OD**	**Glove**	0.91	0.87	0.66	0.87	0.91	0.91	0.92	0.88	0.87
CCRA	**SMOTE**	**tf-idf**	0.97	0.98	0.96	0.93	0.96	0.96	0.99	0.89	0.90
CCRA	**SMOTE**	**cbow**	0.76	0.11	0.81	0.87	0.82	0.89	0.89	0.84	0.86
CCRA	**SMOTE**	**skg**	0.77	0.13	0.81	0.90	0.85	0.94	0.93	0.90	0.91
CCRA	**SMOTE**	**Glove**	0.77	0.52	0.80	0.93	0.81	0.94	0.95	0.90	0.90
CCRA	**BSMOTE**	**tf-idf**	0.97	0.98	0.96	0.93	0.98	0.98	0.99	0.93	0.92
CCRA	**BSMOTE**	**cbow**	0.79	0.12	0.86	0.93	0.83	0.94	0.93	0.89	0.88
CCRA	**BSMOTE**	**skg**	0.80	0.12	0.84	0.94	0.85	0.96	0.97	0.94	0.93
CCRA	**BSMOTE**	**Glove**	0.81	0.52	0.83	0.93	0.85	0.94	0.96	0.90	0.90
CCRA	**ADASYN**	**tf-idf**	0.96	0.97	0.97	0.89	0.90	0.96	0.98	0.89	0.80
CCRA	**ADASYN**	**cbow**	0.72	0.11	0.76	0.78	0.73	0.78	0.84	0.71	0.76
CCRA	**ADASYN**	**skg**	0.71	0.13	0.78	0.84	0.80	0.90	0.87	0.83	0.82
CCRA	**ADASYN**	**Glove**	0.72	0.52	0.76	0.78	0.78	0.86	0.90	0.88	0.78
PCA	**OD**	**tf-idf**	0.91	0.91	0.85	0.85	0.91	0.91	0.89	0.90	0.89
PCA	**OD**	**cbow**	0.91	0.90	0.81	0.84	0.91	0.90	0.92	0.92	0.89
PCA	**OD**	**skg**	0.91	0.91	0.75	0.85	0.91	0.92	0.89	0.89	0.90
PCA	**OD**	**Glove**	0.91	0.91	0.79	0.83	0.91	0.90	0.91	0.87	0.89
PCA	**SMOTE**	**tf-idf**	0.67	0.10	0.78	0.87	0.84	0.88	0.90	0.79	0.81
PCA	**SMOTE**	**cbow**	0.36	0.11	0.84	0.88	0.84	0.88	0.91	0.79	0.81
PCA	**SMOTE**	**skg**	0.38	0.11	0.82	0.90	0.85	0.93	0.94	0.91	0.90
PCA	**SMOTE**	**Glove**	0.29	0.11	0.79	0.86	0.84	0.90	0.94	0.81	0.85
PCA	**BSMOTE**	**tf-idf**	0.71	0.09	0.83	0.90	0.88	0.91	0.93	0.86	0.88
PCA	**BSMOTE**	**cbow**	0.41	0.10	0.87	0.92	0.86	0.93	0.94	0.86	0.87
PCA	**BSMOTE**	**skg**	0.53	0.10	0.86	0.93	0.85	0.94	0.94	0.91	0.90
PCA	**BSMOTE**	**Glove**	0.31	0.10	0.82	0.89	0.88	0.93	0.95	0.86	0.88
PCA	**ADASYN**	**tf-idf**	0.53	0.11	0.73	0.82	0.77	0.85	0.84	0.78	0.76
PCA	**ADASYN**	**cbow**	0.36	0.11	0.77	0.81	0.77	0.80	0.86	0.70	0.73
PCA	**ADASYN**	**skg**	0.39	0.12	0.77	0.80	0.80	0.84	0.89	0.77	0.82
PCA	**ADASYN**	**Glove**	0.28	0.10	0.71	0.83	0.76	0.83	0.86	0.78	0.72

5.1 Word Embedding Techniques

Figure 3 shows the performance values, i.e., Accuracy and AUC of the models developed using text metrics generated by applying different word embedding techniques as input using Box-plot diagrams. Table 2 shows the descriptive statistics for different word embedding techniques used in this work. From Fig. 3 and Table 2, we infer that the performance of the model developed using the text metrics generated by applying the Tf-IDF technique as input is showing better performance, with 91.93 mean, 96.44 median, 91.45 Q1, and 98.07 Q3 accuracy values. On the other hand, the model developed with the CBOW technique's metrics shows the worst performance with a mean accuracy value of 83.21. Further, We used Wilcoxon signed-rank test for statistically comparing the performance of the various web service anti-pattern prediction models developed using text metrics generated by applying different word embedding techniques as input and the original metrics. The null hypothesis considered in this paper is: "The web service anti-pattern prediction model trained using text metrics generated by applying different word embedding techniques as input are not significantly different". The considered null hypothesis is accepted if calculated p-values as ≥ 0.05. Table 3 shows the p-value obtained for the models developed using metrics generated by applying various word embedding techniques. A closer look at Table 3 showed that most of the comparison points have the p-value as '0', i.e., the considered hypothesis is rejected. Hence, we conclude a significant difference between the models' performance developed using the text metrics generated by applying different word embedding techniques as input.

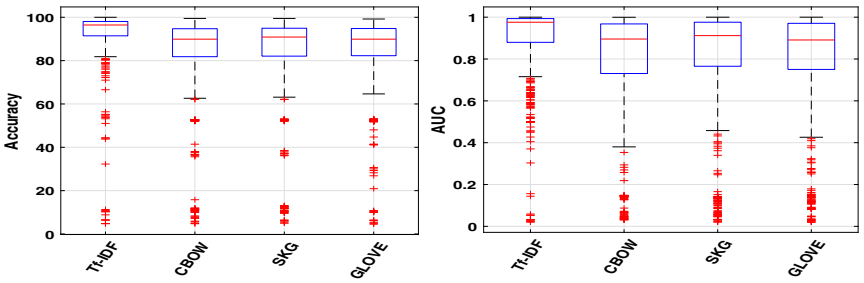

Fig. 3. Box-plot for word embedding techniques

Table 2. Descriptive statistics: word embedding technique

	Min	Max	Mean	Median	Q1	Q3
tf-idf	4.79	100.00	91.93	96.44	91.45	98.07
cbow	4.79	99.47	83.21	89.90	81.82	94.72
skg	5.05	99.46	83.73	90.88	82.07	94.95
Glove	4.52	99.20	84.26	89.90	82.27	94.81

Table 3. Statistical hypothesis: word embedding technique

	tf-idf	cbow	skg	Glove
tf-idf	1	0	0	0
cbow	0	1	0	1
skg	0	0	1	0
Glove	0	1	0	1

5.2 Data Sampling Techniques

Figure 4 shows the performance values, i.e., Accuracy and AUC of the models developed using the data after applying the data sampling techniques along with the original data using Box-plot diagrams. Table 4 shows the descriptive statistics of various data sampling techniques used in this study. From Fig. 4 and Table 4, we conclude that the performance of the models developed after applying data sampling techniques is better when compared to the performance of the models developed using the original data. We also observed that the model developed using SMOTE shows the best performance value with the mean accuracy value of 85.95. On the other hand, the model developed using actual data (OD) delivers the worst performance with the mean accuracy value of 79.71. Further, We used Wilcoxon signed-rank test to statistically compare the performance of the web service anti-pattern prediction models developed using the data after applying various data sampling techniques and the original data. The null hypothesis considered in this paper is: "The web service anti-pattern prediction model trained using the data after applying various data sampling techniques are not significantly different". The considered null hypothesis is accepted if calculated p-values as ≥ 0.05. Table 5 shows the p-value obtained for the models developed using data after applying various data sampling techniques. From Table 5, We observed that most of the comparison points have the p-value of '0'. Therefore, we conclude that the null hypothesis is rejected and that there is a significant difference between the performance of the models developed using the data after applying various data sampling techniques.

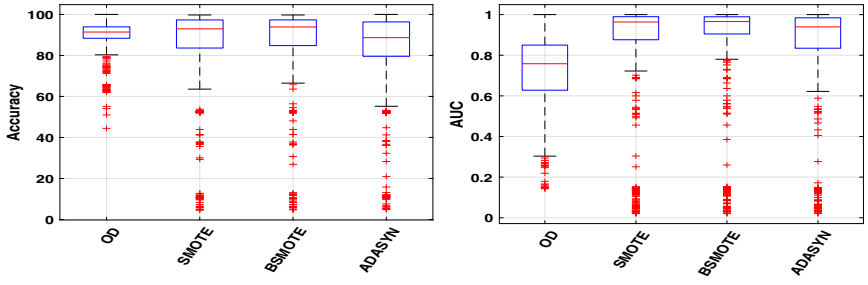

Fig. 4. Box-plots for data sampling techniques

Table 4. Descriptive statistics: data sampling techniques

	Min	Max	Mean	Median	Q1	Q3
OD	44.44	100.00	79.71	88.41	78.38	93.94
SMOTE	4.52	99.73	85.95	94.01	84.96	97.41
BSMOTE	4.79	99.73	85.80	93.88	84.81	97.33
ADASYN	4.80	100.00	82.52	88.66	79.59	96.34

Table 5. Statistical hypothesis: data sampling techniques

	OD	SMOTE	BSMOTE	ADASYN
OD	1	0	0	0
SMOTE	0	1	1	0
BSMOTE	0	1	1	0
ADASYN	0	0	0	1

5.3 Feature Selection Techniques

Figure 5 depicts the performance values, i.e., Accuracy and AUC of the models developed using the features selected by different feature selection techniques as input using Box-plot diagrams. Table 6 shows the descriptive statistics for the various feature selection techniques used in this study. Table 6 show that the mean accuracy value of the model developed using the features selected as significant features (SIGF) using various feature selection and ranking techniques as input is higher than the models developed using the features selected by CC_ra, and PCA as input. Furthermore, Fig. 5 shows that the inter-quartile range for the AUC value for the model generated using PCA is comparatively tall compared to the other models. This indicates that the performance parameters obtained using multiple executions in PCA are exhibiting more variation when compared to other models. Next, we compare the models' performance using the features selected by different feature selection techniques by using the Wilcoxon signed-rank test. The null hypothesis considered here is: "The performance of the anti-pattern prediction models developed using features selected by different feature selection techniques are not significantly different". The defined null-hypothesis is accepted if the p-value obtained using the Wilcoxon signed-rank test is ≥ 0.05. Table 7 shows the p-values of the models developed using various combinations of features as input. From Table 7, we observed that most of the comparison points have the p-value as '0,' i.e., the defined null-hypothesis is rejected. Therefore, there is a significant difference between the models' performance utilizing the features selected by using three different feature selection techniques and the original feature set (AF).

5.4 Classifier Techniques

Figure 6 shows the box-plot diagram of the AUC and the Accuracy of the classifier techniques. Table 8 shows the descriptive statistics for the models trained using

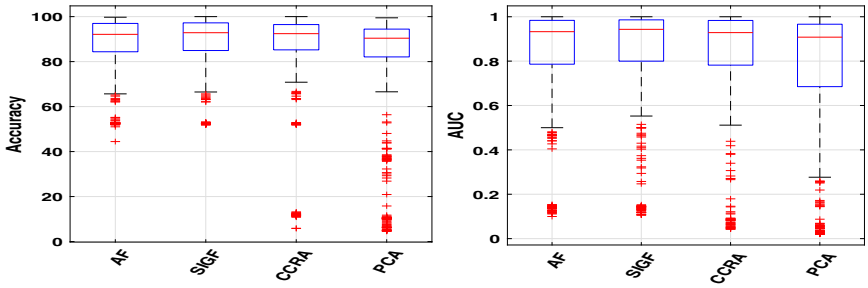

Fig. 5. Box-plots for feature selection techniques

Table 6. Descriptive statistics: feature selection techniques

	Min	Max	Mean	Median	Q1	Q3
AF	44.44	99.73	87.93	92.12	84.38	96.97
SIGF	51.80	100.00	88.72	92.85	84.93	97.22
CCRA	5.85	100.00	86.89	92.42	85.22	96.46
PCA	4.52	99.47	79.60	90.40	82.08	94.44

Table 7. Statistical hypothesis: feature selection techniques

	AF	SIGF	CCRA	PCA
AF	1	0	1	0
SIGF	0	1	0	0
CCRA	1	0	1	0
PCA	0	0	0	1

classifier techniques along with the ensemble techniques. From Table 8 and Fig. 6, we observed that the performance of the model trained using the Extra Trees classifier (EXTR) is higher when compared to the models trained using other classifier techniques. The model trained using the Extra Trees classifier (EXTR) shows good performance with a mean accuracy of 95.35, median accuracy of 96.06, Q1 93.09, and Q3 of 98.36. For Wilcoxon signed rank-sum test, we considered the hypothesis as: "The performance of the anti-pattern prediction models trained using various classifier techniques is not significantly different". The defined null-hypothesis is accepted if the p-value obtained using the Wilcoxon signed-rank test is ≥ 0.05 and is rejected if the p-value is '0'. Table 9 shows the p-values of the models trained using various classifier techniques. From Table 9, we observed that most of the comparison points have the p-value as '0', i.e., the defined null-hypothesis is rejected. Hence we conclude that there is a significant difference between the performance of the models trained using various classifier techniques.

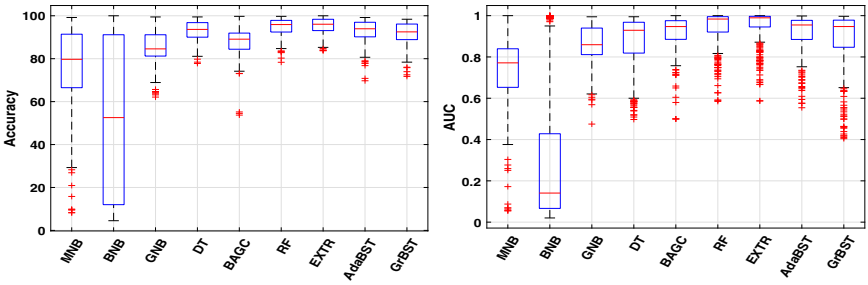

Fig. 6. Box-plots for classifier techniques

Table 8. Descriptive statistics: classifier techniques

	Min	Max	Mean	Median	Q1	Q3
MNB	8.24	99.20	74.47	79.72	66.49	91.41
BNB	4.52	100.00	56.48	52.55	11.99	91.16
GNB	62.12	99.45	85.37	84.57	81.22	91.15
DT	77.78	99.46	92.99	93.65	90.03	96.81
BAGC	53.72	99.73	87.93	89.10	84.41	91.93
RF	78.33	99.73	94.72	95.91	92.42	97.85
EXTR	83.70	100.00	95.35	96.06	93.09	98.36
AdaBST	69.72	99.19	93.00	93.94	90.18	96.97
GrBST	71.75	98.39	91.75	92.50	88.89	96.14

Table 9. Statistical hypothesis: classifier techniques

	MNB	BNB	GNB	DT	BAGC	RF	EXTR	AdaBST	GrBST
MNB	1	0	0	0	0	0	0	0	0
BNB	0	1	0	0	0	0	0	0	0
GNB	0	0	1	0	0	0	0	0	0
DT	0	0	0	1	0	0	0	0	0
BAGC	0	0	0	0	1	0	0	1	0
RF	0	0	0	0	0	1	0	0	0
EXTR	0	0	0	0	0	0	1	0	0
AdaBST	0	0	0	0	1	0	0	1	0
GrBST	0	0	0	0	0	0	0	0	1

6 Conclusion

In this work, we empirically investigated the correlation between the occurrence of anti-patterns and the text metrics. We also investigated the effectiveness of applying four word embedding techniques, three feature selection techniques, three data sampling techniques, and nine classifier techniques to detect web service anti-patterns. Our main findings in this study are:

- Our analysis proved the relationship between anti-patterns in WSDL files and the text metrics generated from the WSDL file.
- We observed that the models trained on data after applying sampling techniques show good performance compared to the models trained on the original data.
- It is also observed that the mean accuracy value of the model developed using the features selected as significant features(SIGF) using various feature selection and ranking techniques as input is higher than the models developed using the features selected by CC_ra, and PCA as input.
- It is observed that the model developed by data after applying the data sampling technique SMOTE shows better performance.
- We observed that the anti-pattern prediction model developed using the features generated by applying the word embedding technique Tf-IDF shows the best performance.
- We observed that the model trained using the ensemble technique Extra Trees classifier is better than the models trained using other classifier techniques.

References

1. Alakuş, C.: Neighborhood construction-based multi-objective evolutionary clustering algorithm with feature selection. Master's thesis (2018)
2. Borovits, N., et al.: Deepiac: deep learning-based linguistic anti-pattern detection in IAC. In: Proceedings of the 4th ACM SIGSOFT International Workshop on Machine-Learning Techniques for Software-Quality Evaluation, pp. 7–12 (2020)
3. Jaafar, F., Guéhéneuc, Y.-G., Hamel, S., Khomh, F., Zulkernine, M.: Evaluating the impact of design pattern and anti-pattern dependencies on changes and faults. Empirical Softw. Eng. **21**(3), 896–931 (2015). https://doi.org/10.1007/s10664-015-9361-0
4. Kumar, L., Sureka, A.: An empirical analysis on web service anti-pattern detection using a machine learning framework. In: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), vol. 1, pp. 2–11. IEEE (2018)
5. Pietrzak, B., Walter, B.: Leveraging code smell detection with inter-smell relations. In: Abrahamsson, P., Marchesi, M., Succi, G. (eds.) XP 2006. LNCS, vol. 4044, pp. 75–84. Springer, Heidelberg (2006). https://doi.org/10.1007/11774129_8
6. Schulte, R.W., Natis, Y.V.: Service oriented architectures, part 1. Gartner, SSA Research Note SPA-401-068 (1996)
7. Segev, A., Toch, E.: Context-based matching and ranking of web services for composition. IEEE Trans. Serv. Comput. **2**(3), 210–222 (2009)

8. Tummalapalli, S., Kumar, L., Bhanu Murthy, N.L.: Detection of web service anti-patterns using machine learning framework. In: Singh, J., Bilgaiyan, S., Mishra, B.S.P., Dehuri, S. (eds.) A Journey Towards Bio-inspired Techniques in Software Engineering. ISRL, vol. 185, pp. 189–210. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-40928-9_10
9. Tummalapalli, S., Kumar, L., Neti, L.B.M.: An empirical framework for web service anti-pattern prediction using machine learning techniques. In: 2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON), pp. 137–143. IEEE (2019)
10. Tummalapalli, S., Kumar, L., Murthy, N.L.B., Krishna, A.: Detection of web service anti-patterns using neural networks with multiple layers. In: Yang, H., Pasupa, K., Leung, A.C.-S., Kwok, J.T., Chan, J.H., King, I. (eds.) ICONIP 2020. CCIS, vol. 1333, pp. 571–579. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-63823-8_65
11. Velioğlu, S., Selçuk, Y.E.: An automated code smell and anti-pattern detection approach. In: 2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA), pp. 271–275. IEEE (2017)