Machine learning-based model categorization using textual and structural features

Machine Learning-based Model Categorization using Textual and Structural Features

Alireza Khalilipour[1[0000-0002-0397-6282]], Fatma Bozyigit[2*[0000-0002-5898-7464]], Can Utku[3[0000-0002-0397-6282]], Moharram Challenger[2[0000-0002-5436-6070]]

[1] Computer Engineering Dept., Islamic Azad University, Qazvin Branch, Iran
alireza.khalilipour@gmail.com
[2] Department of Computer Science, University of Antwerp; and AnSyMo/CoSys corelab, Flanders Make Strategic Research Center, Belgium
{fatma.bozyigit, moharram.challenger}@uantwerpen.be
[3] Department of Computer Engineering, Işık University, Istanbul, Turkey
canutku@outlook.com

Abstract. Model Driven Engineering (MDE), where models are the core elements in the entire life cycle from the specification to maintenance phases, is one of the promising techniques to provide abstraction and automation. However, model management is another challenging issue due to the increasing number of models, their size, and their structural complexity. So that the available models should be organized by modelers to be reused and overcome the development of the new and more complex models with less cost and effort. In this direction, many studies are conducted to categorize models automatically. However, most of the studies focus either on the textual data or structural information in the intelligent model management, leading to less precision in the model management activities. Therefore, we utilized a model classification using baseline machine learning approaches on a dataset including 555 Ecore metamodels through hybrid feature vectors including both textual and structural information. In the proposed approach, first, the textual information of each model has been summarized in its elements through text processing as well as the ontology of synonyms within a specific domain. Then, the performances of machine learning classifiers were observed on two different variants of the datasets. The first variant includes only textual features (represented both in TF-IDF and word2vec representations), whereas the second variant consists of the determined structural features and textual features. It was finally concluded that each experimented machine learning algorithm gave more successful prediction performance on the variant containing structural features. The presented model yields promising results for the model classification task with a classification accuracy of 89.16%.

Keywords: Model Driven Engineering, Model Management, Metamodel, Text Mining, Machine Learning.

1. Introduction

In order to address the challenges and reduce the complexity of software development, one of the key approaches is Model-driven Engineering (MDE), which is a modern software engineering paradigm, focuses on using models as first-class entities. It has gained popularity with academic and industrial communities in software engineering, leading to a plethora of models. Accordingly, intelligent model management is essential for different purposes such as clustering and classification with this number of models. Models are utilized in different engineering and science disciplines, and it is necessary to manage and analyze them using data science techniques to find hidden patterns [1]. Text, audio, image, and video are the forms of data that have drawn a great deal of attention in data science so far. However, model data structure has received little emphasis from the data science.

Due to the specific structure of models, it is not possible to directly use machine learning and data mining algorithms on them. Since many models have textual components, text mining become necessary to handle the texts' implicit structure to perform machine learning algorithms correctly. Text mining includes two basic steps: pre-processing and creating feature sets for data representation. There are two sorts of research commonly applied for text representation, indexing, and term weighting [2]. This paper employed Term Frequency-Inverted Document Frequency (TF-IDF) unsupervised term weighting and word2vec neural language representation methods after the pre-processing for model vectorization. Then, Logistic Regression (LR), Naïve Bayes (NB), k Nearest Neighbors (kNN), Support Vector Machine (SVM), Random Forest (RF), and Artificial Neural Network (ANN) classifiers have been experimented on classification standalone and also together with structural features (number of classes, methods, attributes, and association links, weighted methods and attributes per class, depth of inheritance tree, number of children).

This study contributes to the literature by experimenting with different classifiers with different feature representation strategies. It applies Term Frequency-Inverted Document Frequency (TF-IDF) unsupervised term weighting and word2vec neural language representation methods. Moreover, it is the first attempt to use hybrid feature vector including both textual and structural features for metamodel classification task to the best of our knowledge. As a result, it was concluded that using utilizing machine learning algorithms on hybrid feature vectors significantly improve the classification performance of textual features confirming the motivation point of the paper.

The paper consists of the following sections. Section 2 reviews the literature, whereas Section 3 gives information about the dataset, data pre-processing steps, feature engineering, and methods used in the study. Section 4 presents the details of the experimental study with metrics and results. Finally, Section 5 draws a conclusion.

2. Literature Review

There are many studies implementing model categorization in the literature. It has been observed what kind of information (textual or structural) does the existing methods employ for learning operations. In this respect, previous studies can be classified into two categories, i.e., analysis of textual information and analysis of structural information.

Basciani et al. [3] proposed a hierarchical clustering method for metamodels. Although this paper benefits from a vector-based learning method, the description of a metamodel is considered an ordinary text, and models are compared based on completely textual information for clustering. More advanced techniques inspired by information retrieval and Natural Language Processing (NLP) were employed by Babur [4] to extract features, develop the vector space, and finally evaluate the proposed method through clustering. This study also failed to put sufficient emphasis on the structure of the models. In another study , Babur and Cleophas [5] experimented neural network classifier on a dataset of 555 metamodels.. The experiments were conducted in n-gram states for feature extraction.

Addressing clone detection in metamodels, the paper presented by Babur [6] retrieves the input metamodels for clustering. In every cluster, similar fragments were then sought separately. In other words, an n-gram (n=2) was extracted from the corresponding graph of the metamodel and then stored in vectors. After that, clones were detected through comparisons drawn between those bigrams. Similarly, metamodels were first clustered in the study of Babur et al. [7]; however, subtrees of depth one were used for clone detection. The extracted features contained only the textual information of metamodels and included no graph structures.

Literature search shows that the current studies on model categorization mainly use textual or structural contents separately [23]. Therefore, we used text embedding techniques in combination with kernel-based approaches covering both textual and structural information in the model management to obtain high precision. Overall, the determined features were used to improve machine learning algorithms.

3. Proposed Approach

To address the challenge of using textual information of the models in learning, the proposed approach aims to vectorize to embed the textual information of each model in the resultant vector. The ultimate goal is to enhance accuracy in learning the models developed through machine learning algorithms. This section discusses establishing vectors through models based on textual information in two (related) phases: preprocessing and feature engineering. The Figure 1 illustrates the proposed framework.

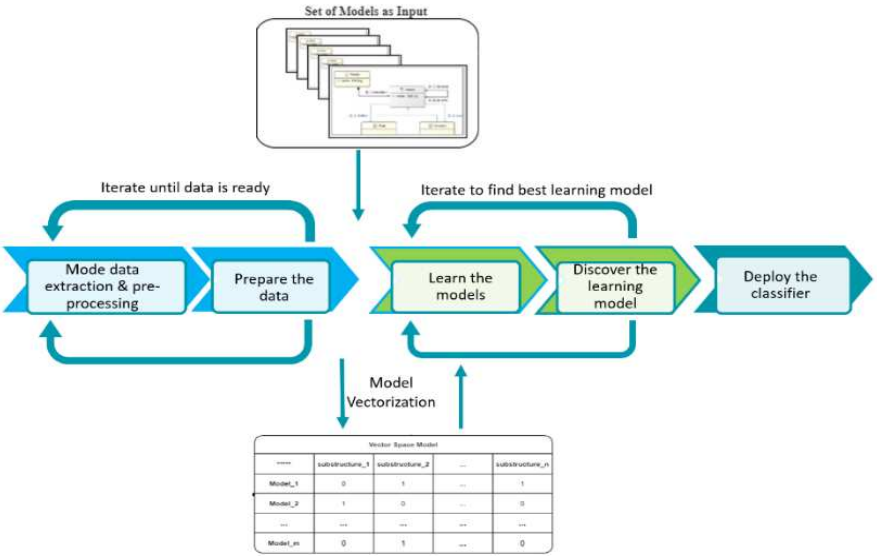

Fig. 1. The proposed approach processes.

3.1. Pre-processing

Step 1: In this step, we formed the raw data in the experimental dataset using PyEcore [8], a MDE framework written for Python. It allows to handle metamodels, the information of classes, references, attributes, and methods were easily obtained. The second step aims to find domain-specific and context-sensitive similarities and semantic relations in the best way possible. Therefore, after lexical refinement through NLP techniques (e.g., tokenization, stemming), the proposed method considers semantic similarity rather than apparent similarity of words. Therefore, WordNet [9], which is an extensive lexical database of English, is used to assess the similarity of words and elements within the related domain. We used WordNet synset instances which are the groupings of synonymous lemmas expressing the same concept, to expand certain lemmas.

By conducting step 1 on the running example (see Fig. 2.), metamodels 1, 2, and 3 of the same domains (the education domain) are detected, whereas Student, Pupil, and Educate elements refer to a common element (Student) in that domain.

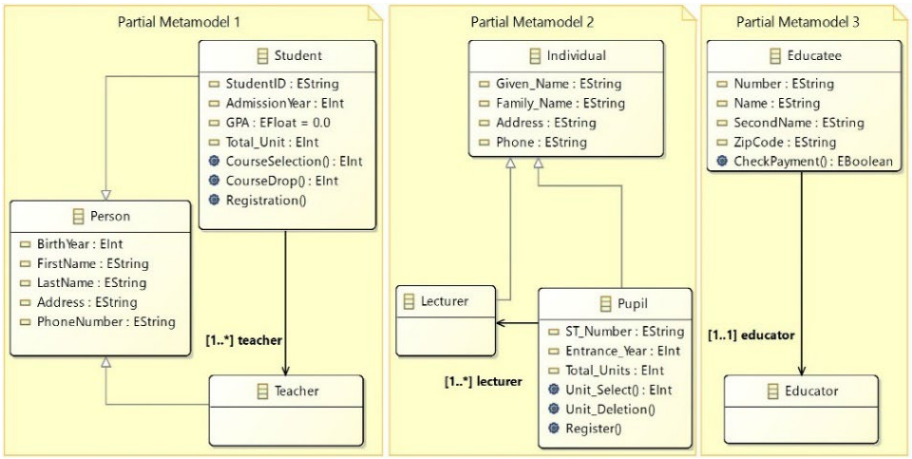

Fig. 2. The partial metamodel examples

Step 2: After detecting semantic similarities, similar elements of the same domain are detected among all models. The common element has a name resembling all similar elements and includes all of the fields existing in common elements (common and uncommon fields). Hence, there are many similar elements in the same-domain models with respect to the common element. They are similar to the common element in different but close ratios. At the end of this step, all common elements of all similar elements are recognized.

In the running example, this step resulted in extracting a common element from three similar elements (Student, Pupil, and Educatee). Figure 3 shows the common element.

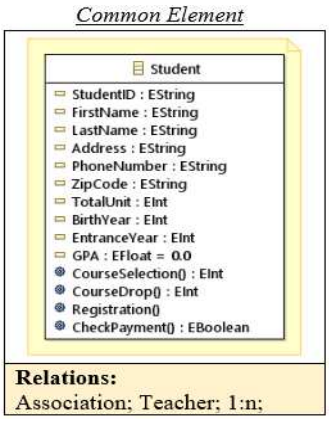

Fig. 3. The common element of Student, Pupil, and Educatee concepts

3.2. Feature Vectorization

Unstructured textual data in models is challenging to process and needs to be described by term sets to represent their contents. The vector space method [10] is one of the most used text representation models for a host of information retrieval operations. This method also appeals to the underlying metaphor of practicing spatial proximity for semantic proximity [11]. There are two sorts of research commonly applied for text representation: indexing and term weighting. Indexing assigns indexing terms for documents, whereas term weighting assigns each term's weight to show its importance. This study uses the word2vec method for indexing and the TF-IDF method to calculate each word's weights in the metamodels.

The word2vec [12], which represents words as continuous vectors, is one of the most common word embedding models. After an external neural network is trained for the word embedding, terms in the document are classified according to their similarities in the word2Vec space. word2vec is basically a neural network-based language modeling method that includes input, output, and hidden layers. It comprises two basic algorithms for training word vectors: continuous word bag (CBOW) and skip-gram. The skip-gram algorithm determines the terms surrounding the target context, whereas the CBOW model predicts by aggregating the distributed representations of the main context. Due to the simple architecture of the CBOW, it works efficiently even with a small amount of training data. However, the skip-gram algorithm provides more efficient results on large datasets than CBOW. The ability to be trained on extensive datasets allows this model to learn complex word relationships such as vec(Ecore) + vec(metamodel) ≅ vec(Eclipse Modeling Framework).

Table 1. Structural features of the models

Structural feature	Description
number of classes	number of classes in the metamodel
number of methods	number of methods in the metamodel
number of attributes	number of attributes in the metamodel
average number of methods	considering a model having i classes and c_i with methods number of m_i $then$ the metric is calculated using: $$ANM = \frac{\sum_0^i m_i}{i}$$
average number of attributes	considering a model having i classes and c_i with attributes of a_i $then$ the metric is calculated using: $$ANA = \frac{\sum_0^i a_i}{i}$$
depth of inheritance tree	depth of a node of tree refers to the length of the maximal path from the node to the root of the tree
number of children	number of immediate descendants of the class
number of association links	number of association relationships of the class
average number of association links	considering a model having i classes and c_i with association relationships of r_i $then$ the metric is calculated using: $$ANR = \frac{\sum_0^i ar_i}{i}$$

TF-IDF is obtained by multiplying the term frequency (TF) and inverse document frequency (IDF) for a term in the text [11]. While TF gives the occurrence frequency of a word in the document, the value IDF (inverse document frequency) indicates this term's occurrence frequency in other documents. The main idea in TF-IDF is to classify terms as much as possible into the same category considering their high appearance in one document and high absence in other documents. When a term appears with a high TF frequency in a text document and rarely appears with low IDF frequency in other documents, it is accepted that the term has a good classification accuracy.
In this paper, we also included the structural metrics during classification process of Ecore diagrams presented in [13] number of classes, number of methods, number of attributes, average number of methods, average number of attributes, depth of inheritance tree, number of children, number of association links, and average number of association links.

3.3. Baseline Machine Learning Algorithms

After the pre-processing and feature selection steps were utilized, some baseline algorithms, commonly used to classify the textual data, were implemented in the first part of this section.
Logistic Regression (LR): LR is used to analyze a data set within one or more independent features [14]. This supervised learning method assigns a new sample to one of the specified discrete classes by employing a logistic function. Logistic regression is a statistical method used to analyze a data set within one or more independent features determining a result.
Naive Bayes (NB): NB depends on the common principle of Bayes Theorem, i.e., a distinct feature in a class is independent of any other feature's presence [15]. It describes the probability of an event based on prior knowledge of conditions. The two main advantages of NB are not requiring a large amount of training data and being able to train comparatively fast than sophisticated models.
k-Nearest Neighbor (kNN): kNN is a supervised learning method that classifies unlabeled observations by assigning them to the label of the most similar k neighbors. The distance metric to find the nearest neighbors of the active instance can be calculated by different methods such as Euclidean, Manhattan, Minkowski, and Hamming.
Support Vector Machine (SVM): SVM trademarked by Vapnik [16] is one of the widely used learning-based pattern classification techniques for classification, regression, and outlier detection. It depends on a solid theoretical background built on statistical learning theory and structural risk minimization techniques. The primary purpose of support vector machines is to find a function in a multidimensional space that can separate the data by a maximal margin.
Random Forest (RF): RF is a commonly used machine learning algorithm merging the output of many classification trees to achieve a single result [17]. The input from samples in the initial dataset is loaded into each decision tree. The prediction of a tree having the most votes is chosen as the outcome. It enables any classifiers with weak correlations to create a robust classifier.

Artificial Neural Network (ANN): ANN is one of the commonly used machine learning algorithms which adopts brain-style information processing consisting of neurons [18]. It has multiple connected layers of nodes with weights and activation functions. The network's processing unit is divided into input, output, and hidden layers. The input layer accepts input data, hidden layers process this instance, and the output layer assembles the result of the system processing result. The power to manage noisy and missing data makes the ANN preferable in data science research.

4. Experimental Study

In this study, we demonstrated our approach on a labeled Ecore metamodel dataset [19] for domain clustering, including 555 models from 9 different categories (bibliography, conference management, bug/issue tracker, build systems, document/office products, requirement/use case, database/sql, state machines, and petri nets). First, data preparation was utilized to transform the raw data (texts in the models) in a useful and efficient format. After pre-processing, two feature representation models (TF-IDF and word2vec) were proposed to observe their effects on categorization accuracy. For the second evaluation, structural features included in textual features and the performance of the machine learning algorithms were observed on the hybrid feature representation. The evaluation results of each method were obtained by dividing the data set into ten pieces by cross-validation.

4.1. Evaluation Metrics

The precision (pr) is obtained by the ratio of the correct data to the total data, and it is calculated according to Equation 1. TP (true positive) denotes the number of objects that are correctly extracted by the system. FP (false positive) refers to the number of objects that the system confirms as true when it is not.

$$precision(pr) = \frac{TP}{TP + FP} \tag{1}$$

The recall (re)metric is calculated by the ratio of the correct data to the expected accurate data given in Equation 2. FN (false negatives) in the equation refers to the number of correct data which could not be found.

$$\text{recall(re)} = \frac{TP}{TP + FP} \tag{2}$$

F1-measure of the proposed approach is defined as the harmonic means of the precision and recall values as demonstrated in Equation 3.

$$F1 - measure = \frac{2 \times pr \times re}{pr + re} \tag{3}$$

4.2. Implementation Details

Firstly, we formed the raw data in the experimental dataset by using PyEcore, an MDE framework written for Python. The machine learning algorithms were designed using the Python Scikit-learn library [20], which provides a high-level construct for classifiers efficiently. It is built on NumPy, SciPy, and Matplotlib. Other Python libraries used in the study are Pandas [21] and Gensim [22]. NumPy is a widely used library for its fast-mathematical computation on arrays and matrices. Pandas library provides rapid, flexible, and expressive data structures. It is an ideal tool for cleaning, modeling, and organizing the analysis results appropriately. Gensim is a Python library generally utilized on large datasets for topic modeling, document indexing, and similarity detection.

4.3. Experimental Results

Table 2 compares the F1-Measure scores obtained using algorithms with TF-IDF and word2vec (CBOW and skip-gram) representations. Considering the experimental results, the highest F1-Measure values were achieved by the ANN classifier in all feature representations. On the other hand, the NB classifier with two representations had the lowest F1-Measure values among all classifiers.

Table 2. Evaluation of different algorithms and feature representations (non-floating)

Algorithm	Feature Model	Representation	Precision	Recall	F1-measure
LR	TF-IDF		0.73	0.76	0.74
	word2vec	skip-gram	0.50	0.66	0.57
		CBOW	0.34	0.58	0.43
NB	TF-IDF		0.80	0.83	0.74
	word2vec	skip-gram	0.53	0.54	0.54
		CBOW	0.59	0.60	0.60
kNN	TF-IDF		0.81	0.78	0.79
	word2vec	skip-gram	0.56	0.58	0.57
		CBOW	0.53	0.54	0.52
SVM	TF-IDF		0.83	0.81	0.82
	word2vec	skip-gram	0.50	0.67	0.57
		CBOW	0.34	0.58	0.43
RF	TF-IDF		0.80	0.82	0.81
	word2vec	skip-gram	0.69	0.74	0.71
		CBOW	0.59	0.66	0.62
ANN	**TF-IDF**		**0.86**	**0.86**	**0.86**
	word2vec	**skip-gram**	**0.70**	**0.75**	**0.72**
		CBOW	0.66	0.69	0.67

Another point to be noticed is that the algorithms with the TF-IDF encoding method performed better than ones with the word2vec method (see Fig. 4). According to our assumption, the poor performances of word2vec feature representations (both cbow and skip-gram) are based on the limited training data. ANN with TF-IDF was evaluated as the most accurate classifier with an 86.45% F1-Measure value. The clos-

est criteria result to ANN were achieved by SVM with TF-IDF with an 82.28% F1-Measure value. On the other hand, TF-IDF-based NB was the worst-performing algorithm with a 74.23% F1-Measure score.

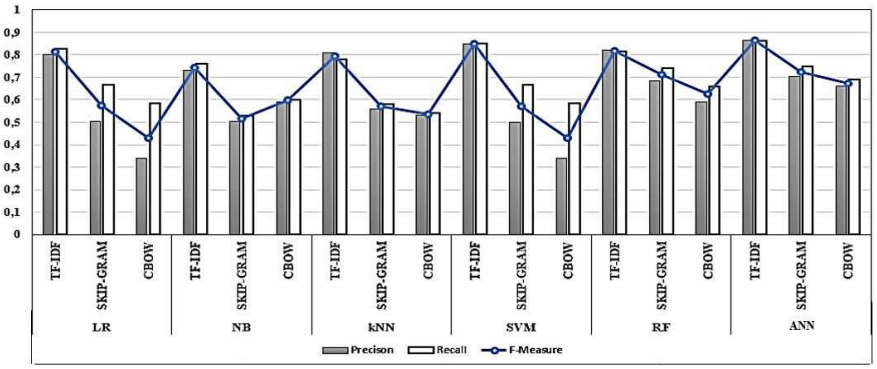

Fig. 4. Experimental results of machine learning classifiers based on TF-IDF and word2vec

The critical point of the experimental study results is that each experimented machine learning algorithms including structural features in addition to the textual features achieved better results on the dataset. For example, using the structural features expressed in Table 1 enhanced the ANN performance by 2.8% (F1-Measure) compared to inputting only textual features. As a result, the study's motivation, "creating hybrid vectors including both textual and structural features could increase the accuracy score of model management." is validated in the second experimented study in the paper (see Table 3).

Table 3. Comparison of textual and hybrid feature representations (non-floating)

Algorithm	**Textual Features (TF-IDF)**			**Textual+Structural Features**		
	Pre	Re	F1-Measure	Pre	Re	F1-Measure
LR	0.73	0.76	0.74	0.76	0.78	0.77
NB	0.80	0.83	0.81	0.82	0.84	0.83
kNN	0.81	0.78	0.79	0.84	0.80	0.82
SVM	0.83	0.81	0.82	0.84	0.85	0.85
RF	0.80	0.82	0.81	0.83	0.84	0.84
ANN	**0.86**	**0.86**	**0.86**	**0.88**	**0.90**	**0.89**

5. Conclusion

Due to a large number of models available nowadays, it is now more necessary than ever to employ intelligent methods to manage these models and their repositories. Using machine learning techniques, intelligent methods can properly meet this need. In this study, the textual information of each model is first summarized in its elements through text processing and NLP techniques, as well as the ontology of synonyms

within a specific domain. The final results were formed as vectors (TF-IDF and word2vec), in which the columns and rows refer to features and models, respectively. Finally, six machine learning classifiers (LR, NB, kNN, SVM, RF, and ANN) with TF-IDF and word2vec feature representations were experimented with to detect models' categories. Experimental results showed that the best-performing method is ANN with TF-IDF weighting scheme, and it achieved 86.45% F1-Measure score. In second experimental study, the performances of machine learning classifiers were observed on different on two different variants of the datasets. The first variant includes only textual features (represented both in TF-IDF and word2vec representations), whereas the second variant consists of the determined structural features and textual features. ANN algorithm was again the most achieved classifier on hybrid vectors with an 89.15% F1-Measure score. It was proved that each machine learning algorithm showed a more successful performance scores in a hybrid feature vector than a pure textual one. The other considerable point is that word2vec based machine learning classifiers showed poor performance in terms of F1-Measure compared to the TF-IDF term weighting scheme. Considering the results, it can be concluded that this study appears promising for future studies on model categorization. As future work, we plan to apply these techniques on industrial models in Model-Driven Engineering [24, 25].

References

1. Tekinerdogan, B., Babur, O., Cleophas, L., Brand, M., and Akşit, M.: Introduction to model management and analytics. Model Management and Analytics for Large Scale Systems, pp. 3-11 (2020).
2. Harish, B. S., Guru, D. S., and Manjunath S.: Representation and classification of text documents: A brief review. International Journal of Computer Applications 2, 110–119 (2010).
3. Basciani, F., Rocco, J. D., Ruscio, D. D., Lovino, L., and Pierantonio, A.: Automated clustering of metamodel repositories. In: International Conference on Advanced Information Systems Engineering, pp. 342-358. Springer, Slovenia (2016).
4. Babur, O.: Statistical analysis of large sets of models. In: 2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 888-891. IEEE, Singapore (2016).
5. Babur, O. and Cleophas, L.: Using n-grams for the Automated Clustering of Structural Models. In: International Conference on Current Trends in Theory and Practice of Informatics, pp 510–524. Springer, Cham (2017).
6. Babur, O.: Clone detection for Ecore metamodels using n-grams. In: MODELSWARD 2018: Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development, pp. 411-219. SciTePress, Portugal (2018).
7. Babur, O., Cleophas, L., and Brand, M.: Metamodel clone detection with Samos. Journal of Computer Languages, vol. 51, pp. 57–74 (2019).
8. Steinberg, D., Budinsky, F., Merks, E., and Paternostro, M.: EMF: Eclipse modeling framework. Pearson Education (2008).
9. Fellbaum, C.: WordNet. In: Theory and applications of ontology: computer applications, pp. 231-243. Springer, Dordrecht (2010).
10. Salton, G., Wong, A., and Yang, C. S.: A vector space model for automatic indexing. Communications of the ACM 18(11), 613–620 (1975).

11. Zhang, W., Yoshida, T., and Tang, X.: A comparative study of TF*IDF, LSI, and multi-words for text classification. Expert Systems with Applications 38(3), 2758–2765 (2011).
12. Church, K. W.: Word2vec. Natural Language Engineering 23(1), 155–162 (2017).
13. Chidamber, S.R., Kemerer, C.F.: A metrics suite for object-oriented design. IEEE Trans. Softw. Eng. 20 (6), 293–318 (1994).
14. Bozyiğit, A., Utku, S., and Nasibov, E.: Cyberbullying detection: Utilizing social media features. Expert Systems with Applications 179, 115001 (2021).
15. Bozyiğit, A., Utku, S., and Nasibov, E.: Cyberbullying detection by using artificial neural network models. In: 2019 4th International Conference on Computer Science and Engineering, pp. 520-524 (2019).
16. Cortes, C. and Vapnik, V.: Support-vector networks. Machine learning 20(3), 273-297 (1995).
17. Basaran, K, Bozyiğit F., Siano, P., Taser, P., and Kilinc, D.: Systematic literature review of photovoltaic output power forecasting. IET Renewable Power Generation 14(19), 3961–3973 (2020).
18. Mishra, M. and Srivastava, M.: A view of artificial neural network. In: 2014 International Conference on Advances in Engineering & Technology Research, pp. 1–3 (2014).
19. Babur, O.: A labeled ecore metamodel dataset for domain clustering, 2019.
20. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., and Vanderplas, J.: Scikit-learn: Machine learning in python. Journal of Machine Learning Research 12, 2825–2830 (2011).
21. McKinney, W.: Pandas: a foundational python library for data analysis and statistics. Python for High Performance and Scientific Computing 14(9), 1-9 (2011).
22. Srinivasa-Desikan, B.: Natural language processing and computational linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras. Packt Publishing Ltd, Birmingham (2018).
23. Khalilipour, A., Bozyigit, F., Utku, C., and Challenger, M.: Categorization of the Models Based on Structural Information Extraction and Machine Learning. International Conference on Intelligent and Fuzzy Systems, pp. 173-181 (2022).
24. Challenger, M., Erata, F., Onat, M., Gezgen, H., Kardas, G.: A model-driven engineering technique for developing composite content applications. 5th Symposium on Languages, Applications and Technologies (SLATE'16), pp. 11:1-11:10 (2016).
25. Asici, TZ., Karaduman, B., Eslampanah, R., Challenger, M., Denil, J., Vangheluwe, H.: Applying Model Driven Engineering Techniques to the Development of Contiki-based IoT Systems. IEEE/ACM 1st International Workshop on Software Engineering Research & Practices for the Internet of Things (SERP4IoT), pp. 25-32 (2019).