SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
VOL. 3, NO. 3, DECEMBER 2020
DOI: 10.35377/saucis.03.03.769969

Sentiment Analysis for Software Engineering Domain in Turkish

Mansur Alp Tocoglu[1]
[1]Manisa Celal Bayar University, Department of Software Engineering, Turgutlu 45400, Manisa, Turkey;
mansur.tocoglu@cbu.edu.tr; +90 236 314 10 10

Received 15 July 2020; Revised 12 November 2020; Accepted 07 December 2020; Published online 30 December 2020

Abstract

The focus of this study is to provide a model to be used for the identification of sentiments of comments about education and profession life of software engineering in social media and microblogging sites. Such a pre-trained model can be useful to evaluate students' and software engineers' feedbacks about software engineering. This problem is considered as a supervised text classification problem, which thereby requires a dataset for the training process. To do so, a survey is conducted among students of a software engineering department. In the classification phase, we represent the corpus by using conventional and word-embedding text representation schemes and yield accuracy, recall and precision results by using conventional supervised machine learning classifiers and well-known deep learning architectures. In the experimental analysis, first we focus on achieving classification results by using three conventional text representation schemes and three N-gram models in conjunction with five classifiers (i.e., naïve bayes, k-nearest neighbor algorithm, support vector machines, random forest and logistic regression). In addition, we evaluate the performances of three ensemble learners and three deep learning architectures (i.e. convolutional neural network, recurrent neural network, and long short-term memory). The empirical results indicate that deep learning architectures outperform conventional supervised machine learning classifiers and ensemble learners.

Keywords: sentiment analysis, software engineering, machine learning, text mining, deep learning

Yazılım Mühendisliği Alanında Türkçe Duygu Analizi

Öz

Bu çalışmanın amacı, sosyal medya ve mikroblog sitelerinde yazılım mühendisliğinin eğitim ve meslek yaşamıyla ilgili yorumların belirlenmesinde kullanılacak bir model sağlamaktır. Bu tür önceden eğitilmiş bir model, öğrencilerin ve yazılım mühendislerinin yazılım mühendisliği hakkındaki geri bildirimlerini değerlendirmek için yararlı olabilir. Bu problem, eğitim süreci için bir veri kümesi gerektiren bir metin sınıflandırma problemi olarak kabul edilmiştir. Veri kümesini oluşturmak için, yazılım mühendisliği bölümü öğrencileri arasında bir anket yapılmıştır. Sınıflandırma aşamasında, geleneksel ve kelime yerleştirme metin gösterme şemalarını kullanılarak ve geleneksel denetimli makine öğrenimi sınıflandırıcıları ve iyi bilinen derin öğrenme mimarilerini kullanılarak doğruluk sonuçları sağlanmıştır. Deneysel analizde, öncelikle beş sınıflandırıcı (Naïve Bayes, k-en yakın komşu algoritması, destek vektör makineleri, rastgele orman ve lojistik regresyon) ile birlikte üç geleneksel metin temsil şeması ve üç N-gram modeli kullanarak doğruluk sonuçları elde edilmiştir. Buna ek olarak, iki ensemble algoritması ve üç derin öğrenme mimarilerinin (convolutional neural network, recurrent neural network, and long short-term memory) performanslarını değerlendirilmiştir. Ampirik sonuçlarda derin öğrenme mimarilerinin geleneksel denetimli makine öğrenimi sınıflandırıcılarından ve ensemble algoritmalarından daha iyi performans gösterdiği tespit edilmiştir.

Anahtar Kelimeler: duygu analizi, yazılım mühendisliği, makine öğrenme, metin madenciliği, derin öğrenme

1. Introduction

In today's world, the enormous quantity of information is generated by users from all over the world with the developments in communication technologies on web. Social networks and microblogging websites are the main sources for people to share commonly for exchanging observations, thoughts, feedbacks and comments about any kind of review. This generated informative data can be in many forms such as image, text, sound, video and so on.

The user-generated text documents include many type of reviews such as product reviews, film reviews, hotel reviews, educational opinion reviews and profession reviews. In all these sources sentiments exist frequently. So it has become popular to extract sentiments out of user-generated text documents. Sentiment analysis in text is a process of identifying and classifying the views of users from text documents into different sentiments, such as, positive, negative and neutral [1]. This extracted informative knowledge can be very useful source to be used for decision support systems and individual decision makers [2].

The feedbacks composed of behaviors and comments of students and professionals about software engineering in social networks and microblogging websites can play an important role to inform people who seeks to find out useful insights about software engineering education and professional life. Thus, the automated extraction of these feedbacks in social networks and microblogging websites becomes a prominent task to accomplished. Sentiment analysis can be employed to find out useful insights to be used for recognizing students' and professionals' feedbacks on education and work life of software engineering.

In this paper, we present a machine learning based approach for sentiment analysis on software engineering students' feedbacks about software engineering education and professional life. To do so, we analyze a corpus composed of 4,896 student reviews in Turkish with the use of conventional text representation schemes for conventional classifiers and word embedding models for deep learning architectures. In the experimental analysis, we use three conventional text representation schemes (i.e., term-presence, term-frequency, TF-IDF) and three N-gram models (1-gram, 2-gram and 3-gram) in conjunction with the five classifiers (i.e., naïve bayes, k-nearest neighbor algorithm, support vector machines, random forest and logistic regression). We also evaluate the classification performances of three ensemble learners (i.e., AdaBoost, Bagging and Voting). In addition, we utilize three deep learning architectures (i.e. convolutional neural network, recurrent neural network, and long short-term memory) using Keras and pre-trained Word2vec word vector representations to compare their predictive performance to conventional machine learning classifiers. To do best of our knowledge, this is the first labeled dataset generated in Turkish for the identification of sentiments of comments about education and profession life of software engineering domain in social media and microblogging sites.

The rest of this paper is structured as follows: In Section 2, related works are presented. In Section 3, the methodology of the study is introduced (namely, dataset collection process, feature engineering and classification algorithms). In Section 4, experimental procedure and the empirical results are presented. Finally, Section 5 concludes the paper and provides a projection for further studies on this topic.

2. Related Works

In literature, sentiment analysis is employed to identify information about software engineering in several areas such as technical contents (issues and commit messages) and crowd-generated contents (forum messages and users' reviews) [3].

Sentiment analysis can be used to extract information from developers' expressions in issues and committed messages. Guzman et al. [4] focused on sentiment extraction of the developer-written comments in GitHub. They found that developers have higher positive comments when they work in projects having more distributed teams. On the contrary, the comments written on Mondays by the developers indicate more negative sentiments. Sinha et al. [5] extracted sentiments from the comments written in 28,466 projects. Based on the results, most of the comments classified as neutral. In addition, the comments written on Tuesdays by the developers indicate more negative sentiments. Ortu et al. [6] used the dataset JIRA [7], composed of 560K issue comments committed by the developers, to analyse the effectiveness of sentiments in comments for issue fixing time. As a result, they found that the issues related the positive comments tend to have shorter fixing time. In contrast, the issues related the negative comments tend to have longer fixing time.

The extraction process of the sentiments from the users' reviews and forum messages play an important role in the evaluation process of the software applications. Goul et al. [8] focused on detecting bottlenecks in requirement engineering by employing sentiment analysis over 5,000 reviews. Carreno et al. [9] used a model unifying aspects and sentiments together to detect topics out of reviews of the applications. In addition, they also extracted users' opinions from the detected topics. Guzman et al. [10] employed SentiStrength [11] for detect topics out of reviews of the applications and extracting users' opinions from the detected topics. Panichella et al. [12] classified users' reviews in three sentiment categories (namely, neutral, positive, and negative) by using Naïve Bayes classifier. In the study [13], the authors focused on analysing sentiments on tweets related to software projects. Calefato et al. [14] presented a sentiment analysis classifier named Senti4SD to be used for extracting the developers' sentiments in communication channels. To do so, first they constructed a dataset from Stack Overflow questions, answers, and comments to be used for the training phase. After that, they manually validated the raw dataset. Senti4SD classifier utilizes from lexicon-based features, keyword-based features and semantic features based on word embedding. In the paper [15], the authors proposed a sentiment joint model framework to be used for analyzing user reviews automatically for product feature requirements evolution prediction. The joint model is constructed by combining supervised Long Short-term Memory based Recurrent Neural Network and unsupervised hierarchical topic model.

In literature, sentiment analysis is also used in several studies for identifying sentiments from Turkish text. Sağlam et al. [16] focused on constructing a sentiment lexicon for Turkish which is composed of 37K words. The new lexicon is tested on a domain independent news dataset and the accuracy performance of the lexicon is calculated as 72.2%. Bayraktar et al. [17] proposed a holistic method to be used in Turkish for aspect-based sentiment analysis. The proposed method is based on statistical, linguistic and rule-based approaches. For evaluation phase, they used a Turkish restaurant dataset which is constructed within the scope of SemEval Aspect Based Sentiment Analysis 2016. They achieved 52.05% accuracy and 56.28% f-score values. Rumelli et al. [18] applied lexicon-based methods and machine learning algorithms together to perform automated sentiment annotation in Turkish text. They achieved 73% accuracy rate as sentiment analysis result of the proposed model. In the study [19], the authors focused on sentiment analysis on Turkish shopping and movie websites. They compared the classification performances of the traditional machine learning algorithms and recurrent neural networks arhitectures. Karcioğlu and Aydin [20] focused on extracting sentiments from Turkish and English twitter posts collected from Twitter. They investigated the performances of BOW and Word2Vec models using Linear Support Vector Machine and Logistic Regression. Ayata et al. [21] applied sentiment analysis on four different sector tweets (namely banking, football, telecom and retail). To do so, they vectorized the datasets with word embedding model and achieved accuracy rates of 89.97%, 84.02%, 73.86% and 63.68% for all sectors in sequence using Support Vector Machine and Random Forests classifiers.

3. Methodology

This section presents the methodology of the study. Namely, the dataset collection, pre-processing, feature extraction, classification algorithms, ensemble learners and deep learning architectures have been briefly presented.

3.1 Dataset Collection and Preprocessing

In this study, we conducted a survey among sophomore and senior students in the Software Engineering Department at Manisa Celal Bayar University in Turkey. In this survey, the students are asked to write five positive and five negative comments for software engineering education and profession life. In total, 349 sophomores and 185 seniors attended in this survey. Here, we assumed the comments obtained from senior students as the source of the knowledge for work life for software engineering since all seniors do direct internship in software companies in the last semester of their education life. As a result of this survey, we managed to create a dataset named Software Engineering Survey Dataset (SESD) (the dataset can be downloaded from http://mansurtocoglu.cbu.edu.tr/). Table 1 shows the distribution of the comments collected among the students. A total of 5,242 documents are collected. 2,614 of these documents are labeled as negative and the rest 2,628 documents are labeled as positive. To validate the dataset, we conducted an annotation process where two annotators annotated each document one by one. As the result of this annotation process, we eliminated 346 documents which are not labeled the same by two annotators. We calculated the Cohen’s kappa (K) metric as 0.97 which indicates a perfect agreement among the annotators [22], [23].

Table 1 The Distribution of the Comments in SESD

# of comments before annotation	**# of comments after annotation**
5,242 (2,614 negative)(2,628 positive)	4,896 (2,306 negative)(2,590 positive)

In the preprocess stage, we preprocessed SESD to use in the classification phase. In the first step, we converted all letters to lowercase and removed all punctuation marks, numeric characters, and extra spaces. Next, we identified stems of each word in the dataset. To do so, we used Snowball-stemmer (SS) algorithm [24]. At last, we removed stop-words by using the Turkish language stopword list provided in Python natural language toolkit [25].

3.2 Feature Extraction Schemes

N-gram modelling is a popular feature representation scheme for language modelling and natural language processing tasks. An n-gram is a contiguous sequence of n items from a given instance of text document. In this scheme, items may be phonemes, syllables, letters, words or characters. In natural language processing tasks, word-based n-grams and character n-grams have been widely utilized. N-gram of size 1 has been referred as “unigram”, N-gram of size 2 has been referred as “bigram” and N-gram of size 3 has been referred as “trigram”. To model comments, we utilized word-based n-gram models, where unigrams, bigrams and trigrams have been taken into consideration.

In the vector space model (VSM), we have considered three different schemes to represent comments, namely, term presence-based representation (TP), term frequency-based representation (TF) and term frequency - inverse document frequency (TF-IDF) based representation. In term frequency-based representation, the number of occurrence of words in the documents have been counted, namely, each document has been represented by an equal length vector with the corresponding word counts. In term presence-based representation, presence or absence of a word in a given document has been utilized to represent text documents. In TF-IDF representation two major equations are multiplied together which are term frequency and inverse document frequency. The inverse document frequency is calculated by taking the logarithm of the equation which is, the number of the total documents within the dataset, divided by the document frequency of the related term.

The conventional text representation schemes, such as TP, TF and TF-IDF, are not able to identify semantic relationships between components in text. In addition, conventional text representation schemes have shortcomings due to high dimensionality and sparsity of feature vector [26]. Recently, neural language models have been successfully employed on natural language processing tasks [27]. In contrast to conventional text representation, neural language models repsesent words in low dimensional spaces by using distributed learning representation [28]. These models focus on capturing similarities between words and providing dense representation of documents with semantic properties with less manual preprocessing.

3.3 Classification Algorithms

In the classification stage, we used both conventional supervised machine learning algorithms (i.e., naïve bayes, k-nearest neighbor algorithm, support vector machines, random forest and logistic regression) and well-known deep learning architectures (namely, convolutional neural network, recurrent neural network and long short-term memory networks).

Naïve Bayes algorithm (NB) is a probabilistic classification algorithm based on Bayes’ theorem. It has a simple structure due to the assumption of conditional independence. Despite its simple structure, it can be effectively utilized in a wide range of applications, including text mining and web mining [29].

Support vector machines (SVM) are supervised learning algorithms that can be utilized to solve classification and regression problems. They can be applied effectively to classify both linear and non-linear data [30]. Support vector machines build a hyperplane in a higher dimensional space to solve classification or regression problem. The hyperplane aims to make a good separation by achieving the largest distance to the nearest training data points of classes (known as functional margin).

Random Forest (RF) is a supervised learning algorithm, combining bagging algorithm and random method of subspace [31]. Decision trees were used as the base learning algorithm in this algorithm. Each tree was constructed based on training data bootstrap samples. A random selection of features was used to provide the variety among the base learners. In response, the model can yield promising learning models on datasets with noisy or irrelevant data.

K-nearest neighbor (KNN) is a learning algorithm for supervised learning tasks, including classification and regression tasks [32]. In this scheme, the class label for an instance has been determined based on the similarity of the instance to its nearest neighbors in the training set. In this scheme, all the instances have been stored and at the time of classification, the class label has been identified based on the examination of the k-nearest neighbors.

Logistic regression (LR) is a supervised learning algorithm. The algorithm provides a scheme to apply linear regression to classication tasks. It employs a linear regression model and transformed target variables have been utilized to construct a linear classication scheme [32].

Convolutional Neural Network (CNN) is type deep neural networks which is widely used in image and video recognition, recommender systems and natural language processing. CNN architecture composes of layers which are embedding, convolution, pooling, flattening and fully connected artificial neural network [33].

Recurrent Neural Network (RNN) is a type of feedforward artificial neural network which can handle variable-length sequence inputs [34]. Unlike traditional feedforward neural networks, RNN uses feedback loops to process sequences in order to maintain memory over time. In the traditional RNN algorithm, recurrent units have very simple structures that have no memory units and additional gates. There is only a simple multiplication of inputs and previous outputs, which is passed through the corresponding activation function. RNN is applicable to tasks of unsegmented, connected handwriting recognition or speech recognition.

Long Short Term Memory (LSTM) is an artificial neural network. Unlike simple RNN, an LSTM recurrent unit contains gates, which are used to maintain memory for long periods of time [35].

3.4 Ensemble Learners

Ensemble learning refers to the process of combing the predictions of multiple supervised learning algorithms and treating the algorithms as a committee of decision makers [36]. Ensemble learning schemes seek to identify a more accurate classification model. In this study, we used the ensembles of the five supervised learning algorithms with two well-known ensemble learning methods which are, AdaBoost and Bagging.

AdaBoost Algorithm

AdaBoost is an ensemble learning algorithm based on boosting [37]. The base learning algorithms were trained sequentially in the algorithm and at each round a new learning model was built. The weight values allocated to misclassified samples will be increased at each round, while the weight values allocated to properly categorized cases will be reduced. In reaction, the algorithm aims to devote more rounds to cases that are more difficult to learn and to compensate for classification mistakes produced in previous models.

Bagging Algorithm

Bagging (Bootstrap aggregating) [38] is another technique of constructing the ensemble. In this system, from the initial training set by bootstrap sampling, distinct training subsets were acquired. The projections produced by the 1 algorithms of base learning were combined with the use of majority voting.

4. Experimental Procedure and Results

In this section, experimental procedure and experimental results have been presented.

4.1 Experimental Procedure

In the empirical analysis, we utilized 10-fold cross validation in all cases. In the empirical analysis, three different feature extraction methods (namely, TF, TP and TF-IDF weighting schemes) and three N-gram models (unigram, bigram and trigram) have been considered in conjunction with five conventional machine learning classifiers (namely, naïve bayes, k-nearest neighbor algorithm, support vector machines, random forest and logistic regression) and two ensemble learners (namely, AdaBoost and Bagging). We also achieved results of Voting ensemble learner by ensembling the four classifiers (namely, SVM, NB, LR and RF). In these experiments, we considered using feature size value as 1,000. In addition, the performances of three well-known deep learning architectures (namely, convolutional neural network, recurrent neural network, and long short-term memory) are compared to each other and to conventional machine learning classifiers. In the experiments based on deep learning architectures, the dataset is represented by using Keras and Word2vec [39] embedding layers. We used the Word2vec pre-trained word representation model which is created from a Turkish corpus named Turkish CoNLL17 with a vocabulary size of 3.6M [40]. We used manual tuning for hyper-parameters for all machine learning algorithms. Unless otherwise stated, we stemmed each term using Snowball-stemmer in all experiments.

Figure 1 shows the comparison of accuracy values of five different machine learning algorithms based on two forms (raw and labeled) of SESD dataset. In the classification process of this empirical experiment, TF-IDF weighting scheme is used as feature extraction method. Regarding the two forms of SESD dataset, the performance of the most classifiers, using labeled form of the dataset, provided higher results than using the raw form of SESD. These results indicate that, annotation process of the SESD dataset increased the results in the empirical experiments as noisy documents are eliminated. Therefore, we used the labeled form of SESD for the remaining experiments.

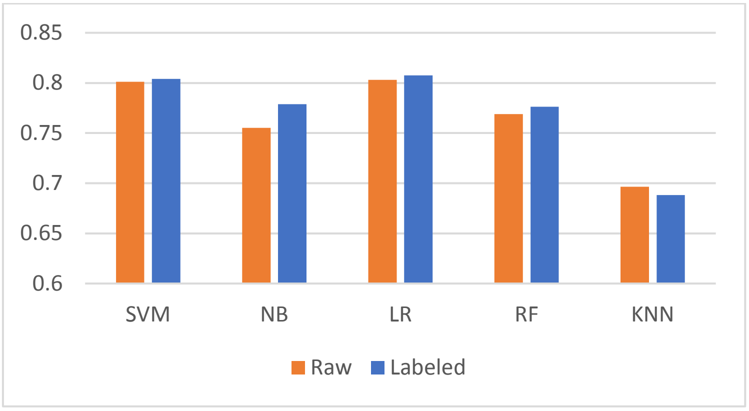

Figure 1 Comparison of Accuracy Values of Five Different Machine Learning Algorithms Based on Two Forms of SESD Dataset

Figure 2 shows the comparison of the accuracy values of five different machine learning algorithms based on the three different conventional text representation schemes (namely, TF-IDF, TF and TP) using unigram model. In all cases, the results obtained by using TF-IDF weighting scheme slightly outperformed the other schemes indicating that considering document frequency of each feature enhance the accuracy performances.

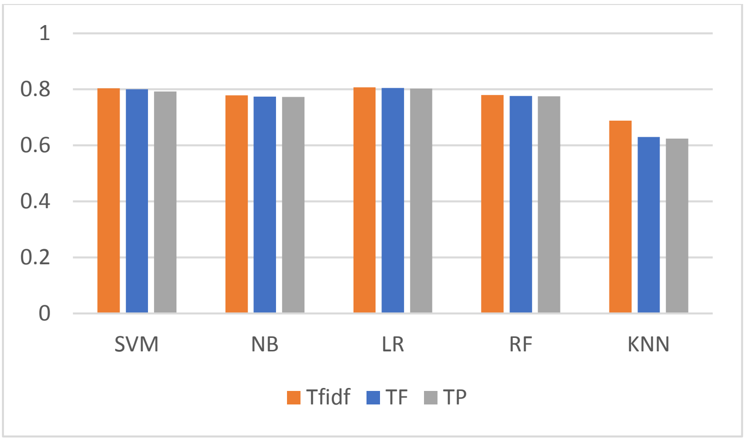

Figure 2 Comparison of the Accuracy Values of Five Different Machine Learning Algorithms Based on Three Different Conventional Feature Extraction Methods

Figure 3 presents the performance comparison of the three N-gram models (namely, unigram, bigram and trigram) in conjunction with five machine learning classifiers. SVM and LR classifiers provided the highest accuracy results among all classifiers in all N-gram models. In contrast, KNN algorithm performed the lowest classification results. Regarding the predictive performance between all three N-gram models, the utilization of unigram for feature set slightly outperformed the other two models. Therefore, we decided to use unigram modeling for the remaining experiments.

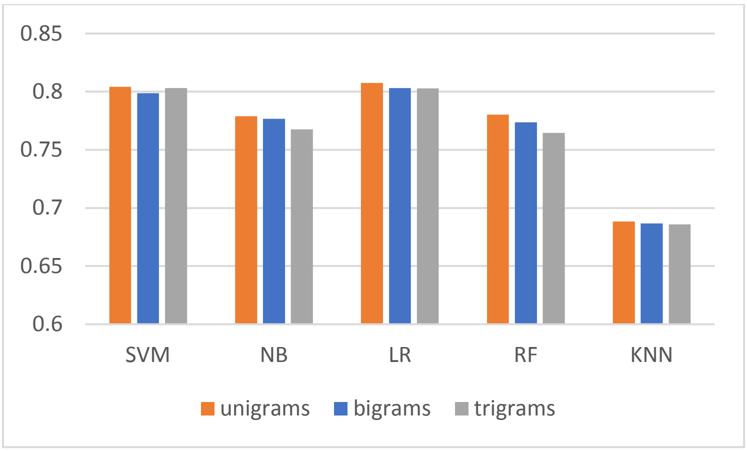

Figure 3 Comparison of Accuracy Values of Five Different Machine Learning Algorithms Based on Three Different N-gram Models

Table 2 shows the classification results of five machine learning algorithms in terms of accuracy, precision and recall measurement values using TF-IDF weighting scheme and unigram feature modeling. We achieved the highest predictive performance as 0.8074 in terms of accuracy by using LR as the classifier. SVM performed the second highest performance with an accuracy value of 0.8041. In contrary, we achieved the lowest accuracy value using KNN with a value of 0.6883. Regarding the precision performance of the classifiers, SVM achieved the highest precision value of 0.8011 which is followed by RF and LR in sequence. Regarding the recall performance of the classifiers, LR slightly outperformed the others with a value 0.8463. SVM achieved the second highest recall value which is followed by NB algorithm. In all cases, KNN algorithm performed the lowest results. This could be due to high dimensional feature size of the dataset and the elimination of the noisy documents.

Table 2 Comparison of Accuracy, Precision and Recall Classification Results of Five Different Machine Learning Algorithms

	SVM	NB	LR	RF	KNN
Accuracy	0.8041	0.7788	**0.8074**	0.7802	0.6883
Precision	**0.8011**	0.7858	0.7950	0.7984	0.7703
Recall	0.8424	0.8027	**0.8463**	0.7766	0.5748

Figure 4 shows the predictive performances of four machine learning classifiers in conjunction with two ensemble learners using unigram features in all cases. As it can be observed from the results presented in Figure 4, we cannot obtain any significant performance enhancement in classification accuracy values with the use of ensemble learners. Beside these results, we also achieved 0.8115 accuracy value by combining the predictions of four machine learning classifiers (namely, SVM, RF, NB and LR) using the voting ensemble learner in which there is also no performance enhancement compared to other two ensemble learners.

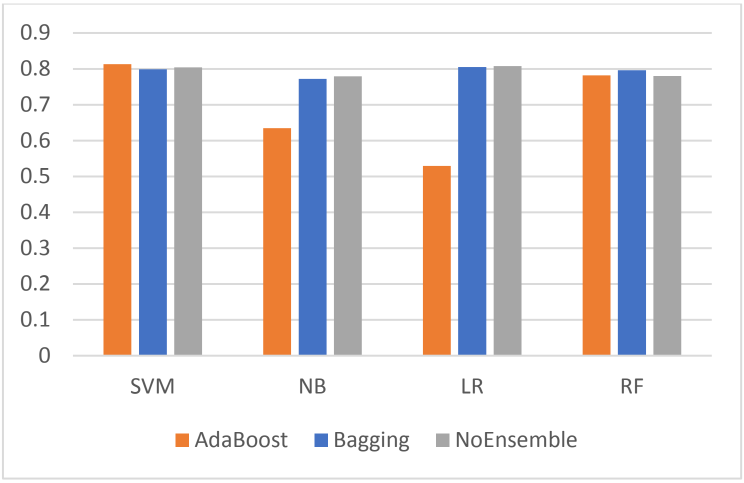

Figure 4. Comparison of Unigram Accuracy Values of Four Different Machine Learning Algorithms Based on Two Ensemble Learners

The performances of the conventional classifiers are extremely related to whether datasets are composed of noisy data labelled by overlapped target classes. However, in this study we eliminated the noisy data from the SESD dataset by conducting an annotation process. This provided a chance for SVM and LR classifiers to achieve higher result compared to others as both of them work relatively well when there is a clear separation between classes. In addition, the higher performances of the classifiers SVM and LR might be due to their high performances against high-dimensional datasets.

Tables 3, 4 and 5 present the accuracy, precision and recall results in sequence which are obtained by using three deep learning architectures (i.e., CNN, LSTM and RNN) with Keras embedding layer in different combination of vector and filter sizes. Regarding the accuracy results, we achieved the highest predictive performance as 0.8315 by using CNN. LSTM performed the second highest performance with a value of 0.8219, which is followed by RNN with 0.8056. Regarding the precision performance of the architectures, CNN achieved the highest precision value of 0.8357 which is followed by LSTM and RNN with results 0.8309 and 0.8070 respectively. For recall performance of the architectures, CNN outperformed others with a value of 0.8587. LSTM achieved the second highest recall value 0.8398 which is followed by RNN with 0.8367. On the other hand, we could not obtain any significant performance enhancement in different combinations of vector and filter sizes.

Generally speaking, CNN outperforms at extracting position invariant features which makes it a better choice for sentiment analysis problems as sentiment extraction is usually based on key phrase. On the other side, RNNs architectures are suitable for problems related to sequence modeling tasks as they require flexible modeling of context dependencies [41]. However, in literature, there is no clear conclusion as there are also studies provided results vice-versa [42], [43]. In brief, the classification results among deep learning architectures depend on the condition of the content of the dataset and the optimization of the parameters of the model.

Table 3 Comparison of Accuracy Classification Results of Three Deep Learning Architectures Based on Different Values of Vector Size and Filter Parameters

Vector size	Filter	CNN	RNN	LSTM
100	100	0.8205	0.7933	0.8219
200	100	**0.8315**	0.7953	0.8211
100	200	0.8256	0.8056	0.8156
200	200	0.8290	0.8017	0.8211

Table 4 Comparison of Precision Classification Results of Three Deep Learning Architectures Based on Different Values of Vector Size and Filter Parameters

Vector size	Filter	CNN	RNN	LSTM
100	100	0.8189	0.7924	0.8275
200	100	0.8294	0.7929	0.8268
100	200	0.8320	0.8070	0.8303
200	200	**0.8357**	0.8068	0.8309

Table 5 Comparison of Recall Classification Results of Three Deep Learning Architectures Based on Different Values of Vector Size and Filter Parameters

Vector size	Filter	CNN	RNN	LSTM
100	100	0.8491	0.8263	0.8398
200	100	**0.8587**	0.8313	0.8382
100	200	0.8402	0.8367	0.8216
200	200	0.8440	0.8255	0.8332

Table 6 shows the accuracy results of the three deep learning architectures (i.e., CNN, LSTM and RNN) using pre-trained word vector representation named Word2vec. Regarding the performances of the architectures, LSTM slightly outperformed others with a value of 0.8572. The results indicate that using Word2vec as an embedding layer compared to Keras provided higher results for all of the three architectures. This could be due to the range of vocabulary in Word2vec model (3.6M).

Table 6 Comparison of Accuracy Classification Results of Three Deep Learning Architectures Using Pre-trained Word2Vec Embedding Layer Based on Different Filter Parameters

Vector size	Filter	CNN	RNN	LSTM
100	100	0.8564	0.8458	0.8550
100	200	0.8564	0.8456	**0.8572**

5. Conclusion

In this paper, we focused on developing a model to be used for identifying sentiments of the comments in software engineering-related social media and microblogging sites as a guidance for people who are willing to learn positive and negative aspects of software engineering education and work life. So this model can be plugged in any application which is implemented for crawling software engineering-related positive and negative information from any text data source.

To generate the dataset, first we conducted a survey to collect labeled documents among software engineering students (349 sophomores and 185 seniors) where we asked them to write five positive and five negative comments about software engineering. Then, we validated the raw dataset with an annotation process which has the Cohen's kappa (K) metric as 0.97 indicating a perfect agreement among the annotators. After the corpus creation phase, we implemented empirical analysis to compare predictive performances of five conventional classifiers (SVM, NB, RF, LR and KNN), three ensemble learners (namely, AdaBoost, Bagging and Voting) and three well-known deep learning architectures (CNN, RNN and LSTM). In addition, we also compared classification results obtained by using different feature extraction methods (namely, TF, TP and TF-IDF) and three N-gram models (unigram, bigram and trigram) in conjunction with conventional classifiers. TF-IDF scheme and unigram model generally outperformed others. In general, the empirical results indicate that SVM and LR classifiers provided the highest predictive performances among other conventional classifiers. This case can be explained with the documents which are clearly separated between classes as a result of the annotation process. Regarding the performances of the ensemble learners, we could not achieve significant performance enhancement using three different ensemble learners on SVM, RF, LR and NB classifiers. Regarding deep learning architectures, CNN provided the highest predictive performance values in almost all compared configurations among other architectures. This might be due to optimization of the parameters of the models and the higher performance of the CNN architecture on sentiment analysis problems as sentiment extraction is usually based on key phrase. Deep learning architectures also performed higher classification results compared to conventional classifiers as they utilize deep neural networks and embedding models. In addition, we also evaluated the proposed deep learning architectures with pre-trained embedding layer Word2vec where we achieved higher accuracy values compared to Keras embedding layer. This case might be due to the range of vocabulary in Word2vec pre-trained model.

For future work, we might extend the size of the dataset by conveying a survey among graduated software engineering students. In addition, different feature extraction methods and embedding schemes can be used to examine performance enhancements.

References

[1] B. Pang and L. Lee, “Opinion mining and sentiment analysis,” *Found. Trends Inf. Retr.*, pp.1–135, 2008.

[2] E. Fersini, E. Messina, and F. A. Pozzi, “Sentiment analysis: Bayesian Ensemble Learning,” *Decis. Support Syst.*, vol. 68, pp.26–38, 2014.

[3] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, M. Lanza, and R. Oliveto, “Sentiment Analysis for Software Engineering: How Far CanWe Go?”, *Proc. - 40th International Conference on Software Engineering,* pp. 94–104, 2018.

[4] E. Guzman, D. Azócar, and Y. Li, “Sentiment Analysis of Commit Comments in GitHub: An Empirical Study,” *Proc. - 11thWorking Conference on Mining Software Repositories*, pp. 352–355, 2014.

[5] M. Goul, O. Marjanovic, S. Baxley, and K. Vizecky, “Managing the Enterprise Business Intelligence App Store: Sentiment Analysis Supported Requirements Engineering,” *Proc. - 45th Hawaii International Conference on System Sciences,* pp. 4168–4177, 2012.

[6] M. Ortu, B. Adams, G. Destefanis, P. Tourani, M. Marchesi, and R. Tonelli, “Are Bullies More Productive? Empirical Study of Affectiveness vs. Issue Fixing Time,” *Proc. - 12th Working Conference on Mining Software Repositories,* pp. 303–313, 2015.

[7] F. Calefato, F. Lanubile, and N. Novielli, “EmoTxt: A Toolkit for Emotion Recognition from Text,” *Proc. - 7th International Conference on Affective Computing and Intelligent Interaction*, pp. 79–80, 2017.

[8] M. Goul, O. Marjanovic, S. Baxley, and K. Vizecky, “Managing the Enterprise Business Intelligence App Store: Sentiment Analysis Supported Requirements Engineering,” *Proc. - 45th Hawaii International Conference on System Sciences,* pp. 4168–4177, 2012.

[9] L. V. G. Carreno and K. Winbladh, “Analysis of User Comments: An Approach for Software Requirements Evolution,” *Proc. - 35th International Conference on Software Engineering,* pp. 582–591, 2013.

[10] E. Guzman, O. Aly, and B. Bruegge, “Retrieving Diverse Opinions from App Reviews”, *Proc. - 9th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement,* pp.21–30, 2015.

[11] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas, “Sentiment in short strength detection informal text,” *J. Am. Soc. Inf. Sci. Technol.,* vol. 61, no. 12, pp. 2544–2558, 2010.

[12] S. Panichella, A. D. Sorbo, E. Guzman, C. A. Visaggio,G. Canfora, and . C. Gall, “How Can I Improve My App? Classifying User Reviews for Software Maintenance and Evolution,” *Proc. - 31st International Conference on Software Maintenance and Evolution,* pp. 281–290, 2015.

[13] E. Guzman, R. Alkadhi, and N. Seyff, “An exploratory study of Twitter messages about software applications,” *Requir. Eng.*, vol. 22, pp. 387–412, 2017.

[14] F. Calefato, F. Lanubile, F. Maiorano, and N. Novielli, “Sentiment polarity detection for software development,” *Empir. Software Eng.,* vol. 23, pp. 1352–1382, 2018.

[15] L. Zhao, and A Zhao, “Sentiment analysis based requirement evolution prediction,” *Future Internet,* vol. 11, no. 2, article no. 5, 2019.

[16] F. Sağlam, H. Sever and B. Genç, “Developing Turkish Sentiment Lexicon for Sentiment Analysis using Online News Media,” *Proc. - 13th International Conference of Computer Systems and Applications,* pp. 1–5, 2016.

[17] K. Bayraktar, U. Yavanoglu and A. Ozbilen, “A Rule-Based Holistic Approach for Turkish Aspect-Based Sentiment Analysis,” *Proc. - IEEE International Conference on Big Data,* pp. 2154–2158, 2019.

[18] M. Rumelli, D. Akkuş, Ö. Kart and Z. Isik, “Sentiment Analysis in Turkish Text with Machine Learning Algorithms,” *Proc. - Innovations in Intelligent Systems and Applications Conference,* pp. 1–5, 2019.

[19] B. Ciftci and M. S. Apaydin, “A Deep Learning Approach to Sentiment Analysis in Turkish,” *Proc. - International Conference on Artificial Intelligence and Data Processing*, pp. 1–5, 2018.

[20] A. A. Karcioğlu and T. Aydin, “Sentiment Analysis of Turkish and English Twitter Feeds Using Word2Vec Model,” *Proc. - 27th Signal Processing and Communications Applications Conference,* pp. 1–4, 2019.

[21] D. Ayata, M. Saraçlar and A. Özgür, “Turkish Tweet Sentiment Analysis with Word Embedding and Machine Learning,” *Proc. - 25th Signal Processing and Communications*

Applications Conference, pp. 1–4, 2017.

[22] A. Onan, “Mining opinions from instructor evaluation reviews: A deep learning approach,” *Comput. Appl. Eng. Educ.*, vol. 28, no. 1, pp. 117–138, 2020.

[23] E. Stamatatos, “A survey of modern authorship attribution methods,” *J. Am. Soc. Inf. Sci. Technol.*, vol. 60, no. 3, pp. 538–556, 2009.

[24] M. F. Porter, “Snowball: A language for stemming algorithms,” 2001.

[25] S. Bird, and E. Loper, “NLTK : The Natural Language Toolkit NLTK : The Natural Language Toolkit,” *Proc. - Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pp. 63–70, 2016.

[26] C. C. Aggarwal and C. X. Zhai, “A survey of text clustering algorithms,” in *Mining Text Data*, pp.77–128, 2012.

[27] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” *Proc. - Advances in Neural Information Processing Systems*, pp. 3111–3119, 2013.

[28] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A Neural Probabilistic Language Model,” 2003. *J. Mach. Learn. Research*, vol. 3, pp. 1137–1155, 2003.

[29] H. Zhang, “The Optimality of Naive Bayes,” *Proc. - 17th International Florida Artificial Intelligence Research Society Conference*, pp. 562–567, 2004.

[30] C. Cortes and V. Vapnik, “Support-Vector Networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[31] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, pp. 5–32, 2001.

[32] M. Kantardzic, *Data Mining: Concepts, Models, Methods, and Algorithms: Second Edition*. Wiley, Hoboken, 2011.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Proc. - 25th International Conference on Neural Information Processing Systems*, pp. 1097-1105, 2012.

[34] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[35] X. Li *et al.*, “Long short-term memory neural network for air pollutant concentration predictions: Method development and evaluation,” *Environ. Pollut.*, vol. 231, pp. 997–1004, 2017.

[36] A. Onan, S. Korukoğlu, and H. Bulut, “Ensemble of keyword extraction methods and classifiers in text classification,” *Expert Syst. Appl.*, vol. 57, pp. 232–247, 2016.

[37] Z.H. Zhou, *“Ensemble Methods: Foundations and Algorithm,”* UK: CRC Press, 2012.

[38] L. Breiman, “Bagging predictors,” *Mach. Learn.*, vol. 24, pp. 123–140, 1996.

[39] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” arXiv preprint arXiv:1301.3781, 2013.

[40] NLPL word embeddings repository, “word embeddings repository homepage,” 2017. [Online]. Available: http://vectors.nlpl.eu/repository/. [Accessed: 25-Nov-2020].

[41] W. Yin, K. Kann, M. Yu, and H. Schutze, “Comparative study of CNN and RNN for natural language processing,” arXiv preprint arXiv:1702.01923, 2017.

[42] D. Tang, B. Qin, and T. Liu, “Document Modeling with Gated Recurrent Neural Network for Sentiment Classification,” *Proc. - Conference on Empirical Methods in Natural Language Processing*, pp. 1422–1432, 2015.

[43] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, “Language Modeling with Gated Convolutional Networks,” arXiv preprint arXiv:1612.08083, 2016.

MANSUR ALP TOCOGLU received the B.Sc. degree in Software Engineering, and the M.Sc. degree in Artificial Intelligent Systems from Izmir University of Economics, Izmir, Turkey, in 2008 and 2013, respectively. In 2018, he received the Ph.D. degree in computer engineering from Dokuz Eylul University, Izmir, Turkey. He is currently an Assistant Professor in Software Engineering Department in Manisa Celal Bayar University, Manisa, Turkey. His research interests include information extraction from text using machine learning techniques.