BERT-Based Sentiment Analysis: A Software Engineering Perspective⋆

Himanshu Batra, Narinder Singh Punn, Sanjay Kumar Sonbhadra, and Sonali
Agarwal

Indian Institute of Information Technology Allahabad, India
{mit2019119,pse2017002,rsi2017502,sonali}@iiit.ac.in

Abstract. Sentiment analysis can provide a suitable lead for the tools
used in software engineering along with the API recommendation sys-
tems and relevant libraries to be used. In this context, the existing tools
like SentiCR, SentiStrength-SE, etc. exhibited low f1-scores that com-
pletely defeats the purpose of deployment of such strategies, thereby
there is enough scope for performance improvement. Recent advance-
ments show that transformer based pre-trained models (e.g., BERT,
RoBERTa, ALBERT, etc.) have displayed better results in the text classi-
fication task. Following this context, the present research explores differ-
ent BERT-based models to analyze the sentences in GitHub comments,
Jira comments, and Stack Overflow posts. The paper presents three dif-
ferent strategies to analyse BERT based model for sentiment analysis,
where in the first strategy the BERT based pre-trained models are fine-
tuned; in the second strategy an ensemble model is developed from BERT
variants, and in the third strategy a compressed model (Distil BERT) is
used. The experimental results show that the BERT based ensemble ap-
proach and the compressed BERT model attain improvements by 6-12%
over prevailing tools for the F1 measure on all three datasets.

Keywords: Sentiment analysis · Transformer · Deep learning · Software
engineering · BERT.

1 Introduction

Sentiment analysis is a technique to classify sentences to positive, negative or
neutral based upon the point of view expressed through that text. Studies reveal
that there are several applications of sentiment analysis for software engineering
(SE) like analysing developers' opinions through Stack Overflow posts, the scope
of improvement in a library from the comments on GitHub issues section [26],
tracking negative comments to inspect bugs in API [33].

The existing research shows that off-the-shelf sentiment analysis tools [18,20]
when tested on SE domain-specific texts tend to disagree with each other. Lin et
al. [15] reported results of five tools: SentiStrength, NLTK, Stanford CoreNLP,
SentiStrength-SE, Stanford CoreNLP SO on app reviews, Stack Overflow and

---

⋆ All authors have contributed equally.

JIRA issues dataset, with the f1-scores ranging from 0.15 to 0.5. SEntiMoji [5], based upon emoji labelled posts, reported an f1-score of 0.47 for the positive sentences and 0.64 for the negative sentences. But still, the utility of the prevailing tools remains quite low for software engineering applications.

The present paper focuses on the scope of improvement of the existing work for the software engineering domain through sentiment analysis. Specifically leveraging the current state-of-the-art performance of BERT (Bidirectional Encoder Representations from Transformers), which has been widely used for major NLP tasks like sentiment analysis, question answering, translation, etc.

1.1   BERT

BERT [6] was introduced in 2018 by Devlin et al., for language modelling, based on bidirectional training of transformer (attention model) [31]. It uses the attention mechanism that gets the actual context of the text and has two important parts: the encoder (to get the input text) and the decoder (that produces the output or the prediction of the task), through which the pre-training and fine-tuning is carried out as shown in Fig 1. There are two major techniques to train the BERT model:

- *Masked language modelling*: This technique is used to mask 15% of the words randomly in a sentence and the model tries to predict the masked word based upon the context of the other (non-masked) words in the sentence.
- *Next sentence prediction*: In this, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. During training, 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document, while in the other 50% a random sentence from the corpus is chosen as the second sentence.
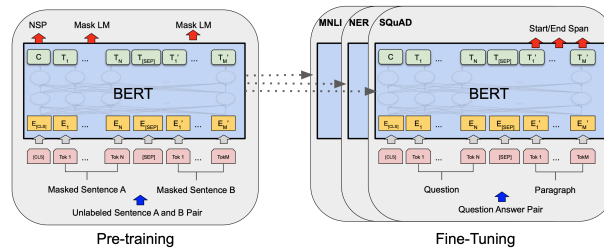

Fig. 1. Pre-training and fine-tuning procedures for BERT [6].

Insufficiency of task-specific labeled data can be handled by BERT through pre-training. This pre-training is done on a large corpus of text (∼2500 million words), which forms a common architecture for a generic task (sentiment analysis, question answering, translation, etc.). This model can then be fine-tuned for

our downstream task, to analyse sentiments. On sentiment treebank dataset [28] BERT attained a GLUE score of 94.9, considered as a benchmark dataset, it has fine-grained sentiment labels for 215,154 phrases in the parse trees of 11,855 sentences. Following the state-of-the-art potential of BERT model, the present research proposes to exploit the features of BERT variants with the ensemble approach that uses three variants of BERT i.e, the base BERT model, RoBERTa and ALBERT. Furthermore, a compressed BERT model (Distil BERT) is used to establish more robust results for sentiment analysis from the perspective of software engineering.

With the objective and contributions highlighted above, the rest of the paper is organized into several sections. The literature review in section 2 presents the recent developments in the deep learning classification task approaches followed by the methods in section 3 to cover the background knowledge and proposed methodology. The later section 4 covers the exhaustive experimental trials over multiple datasets followed by the results in section 5. Finally, the concluding remarks are presented with future research directions in section 6.

2   Related work

With the advent of advancements in the deep learning algorithms, many state-of-the-art frameworks are developed across various domains such as healthcare, natural language processing, target feature learning, etc. [24,25,27,30]. In consideration of sentiment analysis for software engineering domain, [8,9,11], mainly focus on deriving developers' opinions/emotions, along with the context. It is observed that the existing tools make dataset driven predictions, where each prediction conflicts with one another [21]. Table 1 shows the techniques used for the tools along with the datasets over which the training was performed.

Table 1. Prevailing tools.

Tool	Dataset used	Technique
SentiStrength [29]	MySpace informal short texts	Lexicon based
SentiStrength-SE [12]	Jira comments	Lexicon based
SentiCR [1]	Oracle code review comments	Supervised learning
Senti4SD [3]	Stack Overflow	Supervised learning
NLTK [18]	Social media texts	Lexicon based
Stanford CoreNLP [20]	Movie reviews	Recursive neural tensor network

The results of NLTK [18] and Stanford CoreNLP [20] were not very promising as they were not trained on SE-specific datasets and lead to diverging conclusions. SentiCR [1] leveraged boosting algorithms and was based on oracle code review comments but only had two classes i.e., negative and non-negative, unlike others classifying on three classes (positive, negative, and neutral). And it provided the highest f1-score among all the datasets. Senti4SD [3] being trained on

the Stack Overflow dataset with a supervised learning approach had the highest f1-score. Studies have reflected that only to a certain extent these SE-specific tools for sentiment analysis agree with each other. To examine the performance of these tools, Novielli et al. [21] performed a benchmark study on standard datasets (Jira and Stack Overflow comments) and to identify potential weak points for improvement and recorded an f1-score of 0.85 on SentiCR and on SentiStrength-SE [12] it was 0.83 for the Stack Overflow dataset.

Along with the ones mentioned in the Table 1, there are two more related tools, Emotxt [4], based on classifying emotions (anger, sadness, love, joy, fear) and is trained on the Jira and Stack Overflow datasets; whereas, DEVA [10] is a dictionary-based approach to detect valence and arousal in the text that can capture individual emotional states (depression, relaxation, excitement, stress, and neutrality). However, in contrast, BERT with task-specific fine-tuning can provide much better results.

Several other convenient techniques help to derive the sentiment from a text. Data augmentation [14] is one of such techniques that have been used for various downstream tasks and some of the data augmentation techniques used widely are - Lexical substitution of words in the text, which is basically thesaurus based substitution (replacing a word with its synonym) and word-embedding substitution, this technique primarily takes the pre-trained word embeddings to consideration for the replacement of any word in a text through the nearest neighbor in the embedding space (Word2Vec [7], GloVe [23], FasText [2], Sent2Vec [22]). Alternatively, masking of a particular word in a sentence is also a great way to predict a word that has the relative context (BERT). Xie et al [32] used a method called back translation for augmentation of unlabelled text on the IMDB dataset for a semi-supervised model which had only 20 labelled examples. Back translation follows translating an English sentence to some other language (say Spanish) and then translating it back to get the new sentence and that sentence is taken as an augmented sentence for training. However, the utility of the prevailing tools remains quite low for software engineering applications. Following this, the present research work exploits the state-of-the-art potential of the BERT variants for sentiment analysis in the software engineering domain.

3   Proposed work

As per the above discussion, it is evident that software engineering sentiment analysis has always been a challenging task. In this context, the present research proposes a robust solution, where initially, the data augmentation is chosen as a pre-processing strategy to aid in training or fine-tuning the BERT variants for efficient results. In the fine-tuned model an additional untrained layer on top of the BERT model is provided for task-specific learning whereas an ensemble approach is proposed that combines the fine-tuned models with a weighted voting scheme. The sentiment analysis is further extended using a compressed model (Distil BERT) to establish robust results.

3.1   Data augmentation

Data augmentation essentially helps to increase the diversity of the dataset that is available for training the model, without actually getting new data or rather scraping data from relevant websites like Stack Overflow, Jira, GitHub, etc. The available techniques used for text data augmentation are as follows:

Lexical based substitution This technique is based on taking into account the synonym of a random word from a given sentence and substituting it using a thesaurus. Similarly, one basic approach is to provide substitution based on word-embeddings, i.e. to replace a word with its nearest neighboring word in the embedding space.

Back translation It is primarily the translation of a sentence to a different language and then translating it back to the original language, where the sentence translated back is likely to use similar words from the language and the context remains unchanged.

3.2   Fine-tuning BERT model

BERT can be used to get some high-quality language features and can be fine-tuned for the classification task performed on the software engineering datasets to achieve promising results. Fine-tuning provides a few advantages like much faster development, lesser epochs to train on, and limited data requirements. With the pre-trained model, an additional untrained layer of neurons is appended, to fine-tune the model with 2-4 training epochs as suggested in the original BERT paper. Following this three BERT variants are fine-tuned using multiple datasets for better comparative analysis of the proposed strategies for sentiment analysis in software engineering.

3.3   Ensemble BERT models

Based on the fact that combining several models could produce a stronger model hence an ensemble technique is utilized to aggregate the performance of the BERT variants. The scope of prediction is much diverse in the case of the ensemble because there might be certain diversified conclusions on each of the models (that are part of the ensemble), as the core functioning is different for each model. Each model has been pre-trained on a different language modelling task like next sentence prediction used in BERT, sentence order prediction used in ALBERT and dynamic masking used in RoBERTa. Hence it is evident that a stronger model is obtained and the final prediction can be fetched through a voting scheme. For the ensemble technique, BERT base model [6], RoBERTa [17], and ALBERT [13] models are used, as shown in Fig 2. This article used a weighted voting scheme, where the confidence score (aggregated) is considered, i.e. the last Softmax layer output, for each model in the ensemble to get the final weighted prediction.
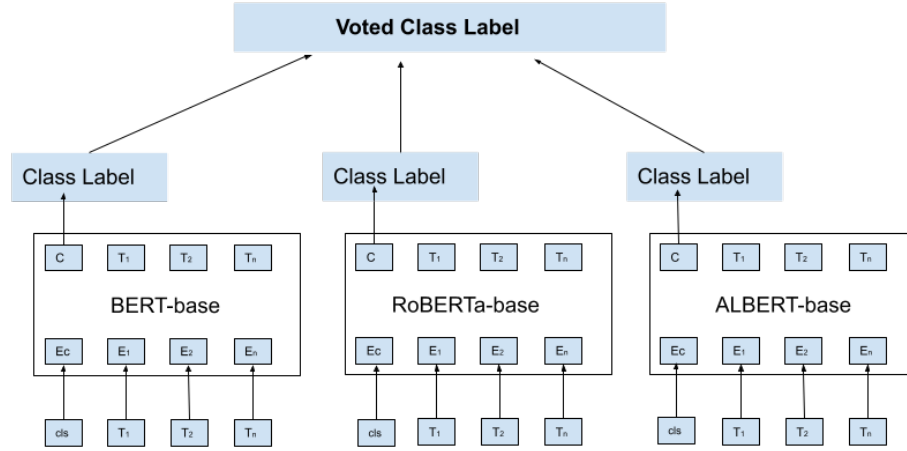
Fig. 2. BERT model variations used for the ensemble technique

3.4 Compressed model

Apart from quantization and model pruning, a distilled model is really helpful to boost up the model's performance rather than to train the model from scratch which requires a large amount of data. The distillation framework mainly contains two models: a larger one and a smaller one. The smaller one mimics the working of the larger one by learning from the probabilities generated by the larger one before the final activation function and produces comparable results. These probabilities are helpful for the generalization capabilities of the compressed model as they are still higher than absolute zero values. Hence, this compressed model is different from the base BERT wherein the model suppresses the lower probability for a particular class and enhances the value for the higher probability class, in its final distribution. The Distil BERT model retains 97% of the language understanding capabilities and the size is reduced by 40% in contrast to BERT base model.

4   Experiments

The experiments are aligned to have an overview of the performance on different datasets [15, 16] and to check: 1) how the model performs after fine-tuning, 2) how well the BERT models perform when used as an ensemble, and 3) to test the performance of the Distil BERT model. The datasets used for the analysis are GitHub commit comments, Jira issue comments, and Stack Overflow posts. Furthermore, a publicly available synthetic dataset was used to check the performance of the model. For the training part, the number of unique sentences

present is 150,000 and the test dataset has 30,000 sentences. It's an added advantage to train and test the model on such a huge dataset, as compared to the Stack Overflow, Jira and GitHub datasets with limited sentences. The complete training part is done in a cloud environment having high-end NVidia GPUs. Fig 3 shows the correlation among the three BERT models used for ensemble strategy. The correlation scores for RoBERTa-ALBERT and ALBERT-BERT is high, which signifies the models have high probability to generate similar predictions in the ensemble approach.
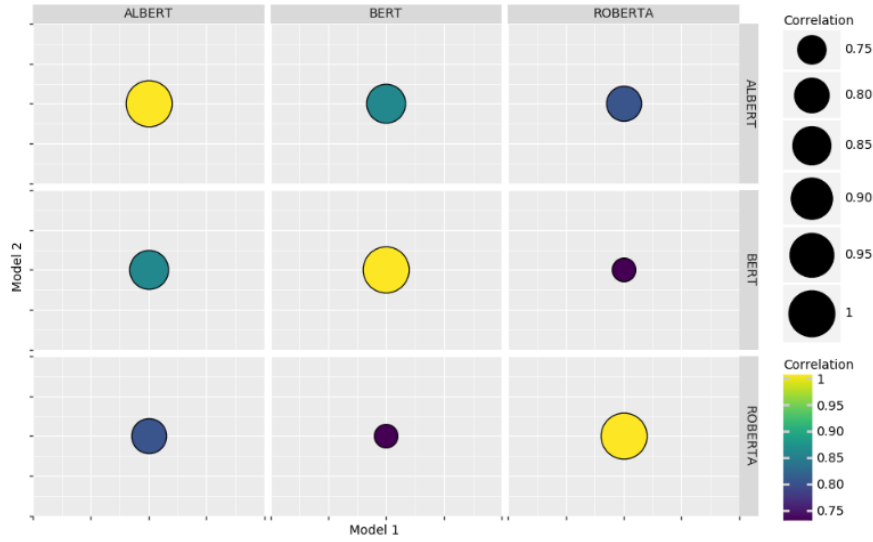

Fig. 3. Correlation plot for the three BERT variants.

Table 2. Train, validation and test split for the three datasets.

Dataset	Train data	Validation data	Test data
Github	4480	498	2137
Stack Overflow	2787	310	1326
Jira	3700	410	1759

4.1 Training and testing

The proposed models are trained and evaluated on the training and test set derived from three datasets as shown in Table 2. The training data is further split into train and validation set for better learning of the model. The training phase

is assisted with the earlystopping technique (halt the training process as soon as the performance stops improving) to avoid the overfitting problem. The trained models are then evaluated on the test set using the standard evaluation metrics, precision, recall and f1-score, for multi-class classification (positive, negative, and neutral). These metrics can be computed using the following equations:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1-\text{Score} = \frac{2 * Recall * Precision}{Recall + Precision} \tag{3}$$

5   Results and discussion

Following from the first proposed strategy involving the fine-tuning of the BERT models, the classification performance concerning sentiment analysis is presented in Table 3. It is observed that from fine-tuning, the lowest f1-score of 0.67 was recorded for the ALBERT model on Stack Overflow dataset. While the RoBERTa model performed better on GitHub and Jira datasets with the f1-scores as 0.91 and 0.83 respectively. The ensemble approach on the other hand was a motivation to obtain a stronger model with slightly varied conclusions from RoBERTa, ALBERT, and BERT-base models, where the f1-score for the GitHub dataset is improved to 0.92, as shown in Table 4.

Table 3. Fine-tuning results for the three available datasets.

Dataset	Model	Neutral			Negative			Positive		
		P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
GitHub	BERT	**0.92**	0.91	**0.91**	0.89	**0.91**	0.90	**0.93**	0.93	**0.93**
	ALBERT	0.85	**0.94**	0.89	0.92	0.80	0.85	0.90	0.93	0.91
	RoBERTa	**0.94**	0.91	**0.92**	**0.90**	0.91	**0.90**	0.91	**0.94**	**0.93**
Stack Overflow	BERT	0.85	0.86	0.85	**0.85**	**0.87**	**0.86**	**0.94**	**0.92**	**0.93**
	ALBERT	0.92	**0.94**	0.93	0.62	0.62	0.62	0.71	0.34	0.48
	RoBERTa	**0.96**	0.92	**0.94**	0.78	0.82	0.80	0.57	0.76	0.65
Jira	BERT	**0.92**	0.89	**0.91**	0.85	0.68	0.76	0.76	**0.94**	0.84
	ALBERT	0.86	**0.94**	0.90	**0.86**	**0.70**	**0.78**	0.78	**0.94**	**0.86**
	RoBERTa	**0.92**	0.90	**0.91**	0.80	0.73	0.76	0.78	**0.88**	0.83

*bold values indicate the highest metric value

The trend with deep learning models is that the training time is directly proportional to the number of parameters. Therefore it is evident that the compressed model can improve the inference time of the model if deployed as a tool. For the Stack Overflow and the Jira datasets, the f1-score improved to 0.88 and

Table 4. Ensemble model results over different datasets.

Dataset	Neutral			Negative			Positive		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
GitHub	0.92	0.93	0.92	0.92	0.90	0.91	**0.93**	**0.93**	**0.93**
Stack Overflow	**0.93**	**0.92**	**0.92**	0.91	0.91	0.91	0.77	0.79	0.78
Jira	0.91	0.90	0.90	0.86	0.65	0.74	0.76	0.92	0.83

*bold values indicate the highest metric value

Table 5. Compressed model results over different datasets.

Dataset	Neutral			Negative			Positive		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
GitHub	0.93	0.92	0.92	0.91	0.92	0.91	**0.93**	**0.93**	**0.93**
Stack Overflow	0.87	0.84	0.86	0.85	0.88	0.86	**0.93**	**0.94**	**0.94**
Jira	0.92	0.89	0.91	0.79	0.77	0.78	0.77	0.89	0.83

*bold values indicate the highest metric value

0.84 respectively, with the Distil BERT model, as shown in Table 5. Additionally, the f1-score recorded for the synthetic data when tested on the ensemble model was 0.85 which proved to be a major improvement.

6   Conclusion

From the perspective of software engineering applications, BERT model shows significant improvement on SE-specific datasets for the classification task when compared to the performance of the prevailing tools. This article proposes that the fine-tuning of the pre-trained model is more effective than taking a large corpus of text and training the model from scratch, while also the ensemble approach can boost the overall performance. The compressed model on the other hand produced similar results and is overall advantageous when deployed as a tool with limited resources. The extensive trials show that the approaches used outperformed the prevailing tools with an improved f1-score of 0.88 and 0.84 for the Stack Overflow and the Jira datasets respectively. Similarly for the GitHub dataset, the f1-score enhanced for positive and negative classes as 0.93 and 0.91 respectively, with an overall f1-score recorded as 0.92. Furthermore, the study shows the results with synthetic data had an overall f1-score of 0.85 on the ensemble model. The paper highlights the improved results based on the BERT model and can further be extended on domain-specific research, while there is still scope to improve these results as the dataset availability is limited concerning the software domain. Bug prediction [19] is another important task that could be studied along with the API recommendation and mining developer's opinions on text/comments.

Acknowledgment

References

1. Ahmed, T., Bosu, A., Iqbal, A., Rahimi, S.: Senticr: A customized sentiment analysis tool for code review interactions (10 2017)
2. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information (2016)
3. Calefato, F., Lanubile, F., Maiorano, F., Novielli, N.: Sentiment polarity detection for software development (2017)
4. Calefato, F., Lanubile, F., Novielli, N.: Emotxt: A toolkit for emotion recognition from text (10 2017)
5. Chen, Z., Cao, Y., Lu, X., Mei, Q., Liu, X.: Sentimoji: An emoji-powered learning approach for sentiment analysis in software engineering (07 2019)
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (10 2018)
7. Goldberg, Y., Levy, O.: word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method (02 2014)
8. Imtiaz, N., Middleton, J., Murphy-Hill, E., Girouard, P.: Sentiment and politeness analysis tools on developer discussions are unreliable, but so are people (06 2018)
9. Islam, M., Zibran, M.: Leveraging automated sentiment analysis in software engineering (05 2017)
10. Islam, M., Zibran, M.: Deva: sensing emotions in the valence arousal space in software engineering text (04 2018)
11. Islam, M., Zibran, M.: Sentistrength-se: Exploiting domain specificity for improved sentiment analysis in software engineering text. Journal of Systems and Software **145** (08 2018)
12. Islam, M., Zibran, M.: Sentistrength-se: Exploiting domain specificity for improved sentiment analysis in software engineering text (08 2018)
13. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: Albert: A lite bert for self-supervised learning of language representations (09 2019)
14. Liesting, T., Frasincar, F., Trusca, M.M.: Data augmentation in a hybrid approach for aspect-based sentiment analysis (2021)
15. Lin, B., Zampetti, F., Bavota, G., Di Penta, M., Lanza, M., Oliveto, R.: Sentiment analysis for software engineering: how far can we go? (05 2018)
16. Lin, B., Zampetti, F., Oliveto, R., Di Penta, M., Lanza, M., Bavota, G.: Two datasets for sentiment analysis in software engineering (09 2018)
17. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach (07 2019)
18. Loper, E., Bird, S.: Nltk: the natural language toolkit (07 2002)
19. Mangnoesing, G.V.H., Trusca, M.M., Frasincar, F.: Pattern learning for detecting defect reports and improvement requests in app reviews (2020)

20. Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., McClosky, D.: The stanford corenlp natural language processing toolkit (01 2014)
21. Novielli, N., Girardi, D., Lanubile, F.: A benchmark study on sentiment analysis for software engineering research (03 2018)
22. Pagliardini, M., Gupta, P., Jaggi, M.: Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. In: NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics (2018)
23. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation (01 2014)
24. Punn, N.S., Agarwal, S.: Chs-net: A deep learning approach for hierarchical segmentation of covid-19 infected ct images. arXiv preprint arXiv:2012.07079 (2020)
25. Punn, N.S., Agarwal, S.: Multi-modality encoded fusion with 3d inception u-net and decoder model for brain tumor segmentation. Multimedia Tools and Applications pp. 1–16 (2020)
26. Rahman, M.M., Roy, C., Kievanloo, I.: Recommending insightful comments for source code using crowdsourced knowledge (09 2015)
27. Rajora, H., Punn, N.S., Sonbhadra, S.K., Agarwal, S.: Web based disease prediction and recommender system (2021)
28. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C., Ng, A., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank (01 2013)
29. Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., Kappas, A.: Sentiment strength detection in short informal text (12 2010)
30. Torfi, A., Shirvani, R.A., Keneshloo, Y., Tavaf, N., Fox, E.A.: Natural language processing advancements by deep learning: A survey. arXiv preprint arXiv:2003.01200 (2020)
31. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., Polosukhin, I.: Attention is all you need (06 2017)
32. Xie, Z., Genthial, G., Xie, S., Ng, A., Jurafsky, D.: Noising and denoising natural language: Diverse backtranslation for grammar correction (01 2018)
33. Zhang, Y., Hou, D.: Extracting problematic api features from forum discussions (05 2013)