JCT at SemEval-2022 Task 6-A: Sarcasm Detection in Tweets Written in English and Arabic using Preprocessing Methods and Word N-grams

Yaakov HaCohen-Kerner, Matan Fchima, Ilan Meyrowitsch

Department of Computer Science, Jerusalem College of Technology, Lev Academic Center
21 Havaad Haleumi St., P.O.B. 16031, 9116001 Jerusalem, Israel
kerner@jct.ac.il, matanf1992@gmail.com, meyrowitsch@gmail.com

Abstract

In this paper, we describe our submissions to SemEval-2022 contest. We tackled subtask 6-A - "iSarcasmEval: Intended Sarcasm Detection In English and Arabic – Binary Classification". We developed different models for two languages: English and Arabic. We applied 4 supervised machine learning methods, 6 preprocessing methods for English and 3 for Arabic, and 3 oversampling methods. Our best submitted model for the English test dataset was an SVC model that balanced the dataset using SMOTE and removed stop words. For the Arabic test dataset, our best submitted model was an SVC model that preprocessed removed longation.

1 Introduction

The rapid development of various types of social networks allows the development of increasingly offensive language in general and sarcasm in particular. Sarcasm is the use of words that mean the opposite of what you want to say especially to insult someone, show irritation, or be funny[1].

Sarcastic language can harm individuals or groups of people and may cause harmful effects on society. Thus, it is important to develop high-quality systems capable of detecting sarcastic expressions automatically.

Sarcasm detection is a difficult task not only for a computer but even for a human being. High-quality performance of this task requires understanding the context of the situation, the relevant culture, and in some cases the specific issue or people involved in this situation (Maynard and Greenwood, 2014).

Moreover, the noisy nature of social media texts especially Twitter messages make the detection task even harder. Therefore, it is an interesting and challenging task to detect sarcasm using supervised machine learning (ML) methods and natural language processing (NLP) tools.

Furthermore, Rosenthal et al. (2014) show a significant drop in sentiment polarity classification performance when processing sarcastic tweets, compared to non-sarcastic ones.

In this paper, we describe our research and participation in subtask 6-A for sarcasm detection in tweets written in two languages: English and Arabic. The full description of task 6 in general and 6-A in particular including the datasets and the participating teams is given in Abu Farha et al. (2022).

The structure of the rest of the paper is as follows. Section 2 introduces a background concerning sarcasm detection, text preprocessing, and text classification with imbalanced classes. Section 3 describes subtask 6-A and its datasets. In Section 4, we present the submitted models and their experimental results. Section 5 summarizes and suggests ideas for future research.

2 Related Research

Most prior textual sarcasm detection datasets have been annotated by using either manual labeling or a weak supervision method. In the first approach, sarcasm labels are provided by human annotators (e.g., Filatova, 2012; Riloff et al., 2013; Abercrombie and Hovy, 2016). However, such labels represent annotator perception, which

[1] www.merriam-webster.com. Retrieved 9-Feb-2022.

Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022), pages 1031 - 1038

may differ from the author intention, as further pointed out by Oprea and Magdy (2020).

In the second approach, texts are labeled as sarcastic if they meet predefined criteria, e.g., including specific tags (e.g. #irony, #sarcastic, #sarcasm) (Ptáček et al., 2014; Khodak et al., 2018). However, this method can lead to noisy labels because of various reasons (Oprea and Magdy, 2020).

To overcome these disadvantages, Oprea and Magdy (2019) introduced another method. In their method, the sarcasm labels for texts are provided by the authors themselves.

Most of the sarcasm detection studies are in the English language (Campbell and Katz, 2012; Riloff et al., 2013; Bamman and Smith, 2015; Rajadesingan et al., 2015; Wallace et al., 2015; Amir et al., 2016; Joshi et al., 2016; Hazarika et al., 2018; Oprea and Magdy, 2019).

There are also some studies in Arabic (Karoui et al., 2017; Ghanem et al.,2019; Abbes et al., 2020; Abu-Farha and Magdy, 2020).

For more information about various issues concerning sarcasm detection please refer to survey papers such as Joshi et al. (2017), Sarsam et al. (2020), Verma et al. (2021), and Moores and Mago (2022).

2.1 Text preprocessing

Text preprocessing is an important step in many NLP domains such as ML, sentiment analysis, text mining, and text classification (TC). In text documents in general and in social text documents in particular, it is common to find various types of noise, e.g., typos, emojis, slang, HTML tags, spelling mistakes, and repetitive letters. Analysis of text that has not been carefully cleaned or preprocessed might lead to misleading results.

Not all of the preprocessing types are considered effective for TC tasks. For instance, HaCohen-Kerner et al. (2008) demonstrated that the use of word unigrams including stop words leads to improved results compared to the results obtained using word unigrams excluding stop words.

HaCohen-Kerner et al. (2019) investigated the impact of all possible combinations of six preprocessing methods (spelling correction, HTML tag removal, converting uppercase letters into lowercase letters, punctuation mark removal, reduction of repeated characters, and stopword removal) on TC in three benchmark mental disorder datasets. In another study, HaCohen-Kerner et al. (2020) explored the influence of various combinations of the same six basic preprocessing methods mentioned in the previous paragraph on TC in four general benchmark text corpora using a bag-of-words representation. The general conclusion was that it is always advisable to perform an extensive and systematic variety of preprocessing methods, combined with TC experiments because this contributes to improving TC accuracy.

2.2 Text classification with imbalanced classes

The problem of TC with imbalanced classes is that there are too few examples of the minority class to effectively learn a good predictive TC model. There are various methods to cope with this problem. The main idea is to change the dataset until a more balanced distribution is reached. Two well-known sampling methods that enable such a change are oversampling and undersampling. Random oversampling means randomly duplicating examples in the minority class. Random undersampling means randomly deleting examples in the majority class.

An additional frequent method is to generate synthetic samples which means randomly sampling the attributes from instances in the minority class. There are several algorithms that support the generation of synthetic samples. The most popular is called the Synthetic Minority Oversampling Technique (SMOTE) (Chawla, 2002). This method is an oversampling method that creates synthetic samples from the minor class instead of creating copies. This method selects two or more similar instances and perturbs an instance one attribute at a time by a random amount within the difference to the similar instances. We used also two other oversampling methods BorderlineSMOTE and ADASYN that work similarly.

Readers interested in expanding and deepening the topic of solutions to TC with imbalanced classes are referred to the following articles (Chawla et al., 2002; He and Ma, 2013; Krawczyk, 2016; Brownlee, 2020, Shaikh et al., 2021).

3 Task and Datasets Description

Tables 1-4 present various statistical details about the training and test sets for English and Arabic**.**

The analysis of the details presented in Tables 1 and 2 show that the training and test sets for English are highly imbalanced, with a ratio of about 25:75 and 14:86 (non-sarcastic:sarcastic), respectively. We tried to balance the dataset in our experiments using the oversampling methods that we have mentioned above.

	Sarcastic	Not sarcastic	Total
Documents	867	2,600	3,467
% Docs	25.008%	74.992%	100%
words	15,863	49,432	65,295
characters	86,932	275,172	362,104
avg word per doc	18.296	19.012	18.8333
avg chars per doc	100.267	105.835	104.443
words std	10.235	11.595	11.275
chars std	57.545	65.504	63.653

Table 1: details of the training set for English.

	Sarcastic	Not sarcastic	Total
Documents	200	1,200	1,400
% Docs	14.286%	85.714%	100%
words	4,455	18,505	22,960
characters	23,828	99,024	122,852
avg word per doc	22.275	15.42	16.4
avg chars per doc	119.14	82.52	87.751
words std	12.687	8.861	9.800
chars std	66.224	48.322	52.841

Table 2: details of the test set for English.

	Sarcastic	Not sarcastic	Total
Documents	1,490	2,357	3,847
% Docs	38.732%	61.268%	100%
words	6,370	36,119	42,489
characters	32,858	205,286	238,144
avg word per doc	8.55	15.324	13.697
avg chars per doc	44.104	87.096	76.771
words std	3.924	6.428	6.593
chars std	21.166	36.797	38.389

Table 3: details of the training set for Arabic.

	Sarcastic	Not sarcastic	Total
Documents	200	1,200	1,400
% Docs	14.286%	85.714%	100%
words	1,454	7,545	8,999
characters	7,430	37,772	45,202
avg word per doc	7.27	6.287	6.427
avg chars per doc	37.15	31.476	32.287
words std	4.190	3.574	3.685
chars std	22.094	19.64	20.107

Table 4: details of the test set for Arabic.

The analysis of the details presented in Tables 3 and 4 show a similar picture. The training and test sets for Arabic are highly imbalanced, with a ratio of about 39:61 and 14:86 (non-sarcastic:sarcastic), respectively. Also, for Arabic, we tried to balance the dataset in our experiments using the oversampling methods that we have mentioned above. Both for English and Arabic, the test datasets are even more imbalanced than the compatible training datasets.

4 The Submitted Models and Experimental Results

We applied 4 supervised ML methods on the training datasets: Random Forest (RF), Support Vector Classifier (SVC), Logistic regression (LR), and Decision Tree (DT).

RF is an ensemble learning method for classification and regression (Breiman, 2001). Ensemble methods use multiple learning algorithms to obtain improved predictive performance compared to what can be obtained from any of the constituent learning algorithms. RF operates by constructing a multitude of decision trees at training time and outputting classification for the case at hand. RF combines Breiman's "bagging" (Bootstrap aggregating) idea in Breiman (1996) and a random selection of features introduced by Ho (1995) to construct a forest of decision trees.

SVC is a variant of the support vector machine (SVM) ML method (Cortes and Vapnik, 1995) implemented in SciKit-Learn. SVC uses LibSVM (Chang & Lin, 2011), which is a fast implementation of the SVM method. SVM classifies vectors in a feature space into one of two sets, given training data. It operates by constructing the optimal hyperplane dividing the two sets, either in the original feature space or in higher dimensional kernel space.

LR (Cox, 1958; Hosmer et al., 2013) is a linear classification model. It is known also as maximum entropy regression (MaxEnt), logit regression, and the log-linear classifier. In this model, the probabilities describing the possible outcome of a single trial are modeled using a logistic function.

DT (Song and Ying, 2015) is a flowchart-like structure method in which each internal node represents a "test" on an attribute (e.g. whether a

coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes).

These ML methods were applied using the following tools and information sources:

- The Python 3.7.3 programming language[2].
- Scikit-learn – a Python library for ML methods[3].
- Numpy – a Python library that provides fast algebraic calculous processing, especially for multidimensional objects[4].

We applied six preprocessing methods for English:

1. change to lower-case
2. stop words removal
3. numbers removal
4. emojis removal
5. HTML tags removal
6. punctuations removal

For Arabic we applied three preprocessing methods:

1. punctuations removal
2. Tashkeel removal[5]
3. longation removal[6,7]

We applied three oversampling methods to balance the data:

4. SMOTE
5. BorderlineSMOTE
6. ADASYN

Tables 5 and 6 present the F1-scores over the PCL class of our models for English and Arabic, respectively. The analysis of the details presented in Tables 5 and 6 show that there is a big impact on the longation in the Arabic language, and removing it improved the models. In the English language, we do not see a specific preprocess method that is the most effective method (mark in bold – the three highest results in each language for different number of word unigrams).

Tf-idf	balance method	model	Train test	preprocessing methods	F1-score
500	SMOTE	SVC	70-30	SW, LC	**0.413**
	SMOTE	SVC	80-20	SW, LC, Only words	0.413
	ADASYN	LR	70-30	SW	0.412
1000	BorderlineSMOTE	LR	80-20	SW, LC, Only words	**0.433**
	ADASYN	SVC	70-30	NONE	0.425
	BorderlineSMOTE	SVC	80-20	SW, LC, Only words	0.424
2000	ADASYN	SVC	80-20	Only words	**0.429**
	SMOTE	SVC	80-20	SW, LC, Only words	0.410
	BorderlineSMOTE	LR	80-20	SW	0.407
3000	BorderlineSMOTE	LR	80-20	SW, LC	0.407
	BorderlineSMOTE	SVC	80-20	SW, LC	0.403
	ADASYN	LR	80-20	SW, LC	0.402
4000	ADASYN	LR	70-30	SW, LC	0.410
	SMOTE	SVC	80-20	SW, Only words	0.408
	BorderlineSMOTE	SVC	80-20	SW, LC, Only words	0.401
5000	SMOTE	LR	80-20	NONE	0.401
	BorderlineSMOTE	LR	70-30	LC	0.4
	SMOTE	SVC	80-20	SW	0.39

Table 5: F1-scores over the PCL class of our models for English.

[2] https://www.python.org/downloads/release/python-373/
[3] https://scikit-learn.org/stable/index.html
[4] https://numpy.org
[5] https://github.com/motazsaad/process-arabic-text
[6] https://github.com/bakrianoo/aravec
[7] https://medium.com/analytics-vidhya/sentiment-analysis-of-arabic-text-data-tweets-4e96c8da892b

Tf-idf	balance method	model	Train test	preprocessing methods	F1-score
500	BorderlineSMOTE	LR	80-20	RL	**0.690**
	ADASYN	SVC	80-20	RL	0.682
	NONE	SVC	80-20	RL	0.679
1000	ADASYN	LR	80-20	RT, RL	**0.679**
	NONE	SVC	80-20	RL	0.679
	NONE	SVC	80-20	RP, RL	0.672
2000	ADASYN	LR	80-20	RL	**0.698**
	ADASYN	SVC	80-20	RL	0.685
	NONE	SVC	80-20	RL	0.679
3000	NONE	SVC	80-20	RL	0.679
	NONE	SVC	80-20	RP, RL	0.672
	NONE	SVC	80-20	RT, RL	0.670
4000	SMOTE	LR	80-20	RT, RL	0.679
	NONE	SVC	80-20	RL	0.679
	ADASYN	SVC	80-20	RP, RT	0.676
5000	NONE	SVC	80-20	RL	0.679
	NONE	SVC	80-20	RP, RL	0.672
	NONE	SVC	80-20	RT, RL	0.670

Table 6: F1-scores over the PCL class of our models for Arabic.

Table 7 presents the F1-Scores over the PCL class of our best models for English and Arabic that were submitted to the competition. The F1-scorer over the PCL class on the training dataset of our best model for English and Arabic were 0.4183 and 0.6791 respectively while the F1-scores over the PCL class on the test dataset of our best model were only 0.215 and 0.29515 respectively.

Possible explanations might be: (1) The training dataset is different in its balance rate than the balance rate of the competition test datasets and (2) the content of a relatively high number of Tweeter messages in the competition test dataset are fundamentally different from the content of the Twitter messages in the training dataset.

5 Summary and Future Research

In this paper, we describe our models and submissions to subtask 6-A of SemEval-2022: sarcasm detection in Twitter messages written in English and Arabic using preprocessing methods and word n-grams.

Language	Model name and split mode	ML Method	Applied text preprocessing and oversampling methods	F1-score over the PCL class on the training dataset	F1-score over the PCL class on the test dataset	Place in the competition
English	Ilan_SVC 80-20	SVC	Balance with SMOTE, removed stop words	0.418	0.215	33-35
Arabic	Matan_SVC 80-20	SVC	No balance method, removed longation	0.679	0.295	24

Table 7: F1-scores over the PCL class of our best models for English and Arabic that were submitted to the competition.

We applied 4 supervised ML methods, 6 preprocessing methods for English and 3 for Arabic, and 3 oversampling methods.

Our best submitted model for the English test dataset was an SVC model that balanced the dataset using SMOTE and removed stop words. For the Arabic test dataset, our best submitted model was an SVC model that preprocessed longation removal.

There are various ideas for future research that are connected to the nature of Twitter messages as follows: (1) the use of skip character n-grams because they serve as generalized n-grams that allow us to overcome problems such as noise and sparse data (HaCohen-Kerner et al., 2017), which are common to Twitter messages and (2) Many Twitter messages contain acronyms. Acronym disambiguation might enable better classification (HaCohen-Kerner et al., 2010A).

Another idea that may lead to better classification is to use additional feature sets such as stylistic feature sets (HaCohen-Kerner et al., 2010B).

References

Ines Abbes, Wajdi Zaghouani, Omaima El-Hardlo, and Faten Ashour. 2020. DAICT: A dialectal Arabic irony corpus extracted from Twitter. In Proceedings of the 12th Language Resources and Evaluation Conference, pages 6265–6271, Marseille, France. European Language Resources Association.

Gavin Abercrombie and Dirk Hovy. 2016. Putting Sarcasm Detection into Context: The Effects of Class Imbalance and Manual Labeling on Supervised Machine Classification of Twitter Conversations. In Proceedings of the ACL 2016 Student Research Workshop, pages 107–113. ACL.

Ibrahim Abu Farha, Silviu Oprea, Steve Wilson, and Walid Magdy. 2022. SemEval-2022 Task 6: iSarcasmEval, Intended Sarcasm Detection in English and Arabic. In the 16th International Workshop on Semantic Evaluation (SemEval-2022), Association for Computational Linguistics.

Silvio Amir, Byron C. Wallace, Hao Lyu, Paula Carvalho, and Mario J. Silva. 2016. Modelling context with user embeddings for sarcasm detection in social media. In CoNLL, pages 167–177. ACL.

David Bamman and Noah A. Smith. 2015a. Contextualized sarcasm detection on twitter. In ICWSM, pages 574–577. AAAI Press.

Leo Breiman. 1996. Bagging predictors. Machine learning 24(2), 123-140.

Leo Breiman. 2001. Random forests. Machine learning 45(1), 5-32.

Jason Brownlee. 2020. Imbalanced classification with python: Better metrics, balance skewed classes, cost-sensitive learning. Machine Learning Mastery.

Chih-Chung Chang, and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. ACM transactions on intelligent systems and technology (TIST) 2(3), 1-27.

Nitesh V. Chawla, Kevin W Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: synthetic minority oversampling technique. Journal of artificial intelligence research, 16, 321-357.

John D. Campbell and Albert N. Katz. 2012. Are there necessary conditions for inducing a sense of sarcastic irony? Discourse Processes, 49(6), 459–480.

Corinna Cortes and Vladimir Vapnik. 1995.Support-vector networks. Machine learning 20.3 : 273-297.

David R. Cox. 1958. The regression analysis of binary sequences, Journal of the Royal Statistical Society: Series B (Methodological), 20, 215–232.

Elena Filatova. 2012. Irony and sarcasm: Corpus generation and analysis using crowdsourcing. In LREC. ELRA.

Bilal Ghanem, Jihen Karoui, F. Benamara, Veronique ´ Moriceau, and P. Rosso. 2019. Idat at fire2019: Overview of the track on irony detection in Arabic tweets. Proceedings of the 11th Forum for Information Retrieval Evaluation.

Yaakov HaCohen-Kerner, Ariel Kass, and Ariel Peretz. 2008. Combined one sense disambiguation of abbreviations. In Proceedings of ACL-08: HLT, Short Papers, Association for Computational Linguistics, pages 61-64, Columbus, Ohio, Association for Computational Linguistics. URL: https://aclanthology.org/P08-2.

Yaakov HaCohen-Kerner, Ariel Kass, and Ariel Peretz. 2010A. HAADS: A Hebrew Aramaic abbreviation disambiguation system. Journal of the American Society for Information Science and Technology, 61(9), 1923-1932.

Yaakov HaCohen-Kerner, Hananya Beck, Elchai Yehudai, and Dror Mughaz. 2010B. Stylistic feature sets as classifiers of documents according to their historical period and ethnic origin. Applied Artificial Intelligence, 24(9), 847-862.

Yaakov HaCohen-Kerner, Ziv Ido, and Ronen Ya'akobov. 2017. Stance classification of tweets using skip char Ngrams. In Joint European

Conference on Machine Learning and Knowledge Discovery in Databases (pp. 266-278). Springer, Cham.

Yaakov HaCohen-Kerner, Yair Yigal, and Daniel Miller. 2019. The impact of Preprocessing on Classification of Mental Disorders, in Proc. of the 19th Industrial Conference on Data Mining, (ICDM 2019), New York.

Yaakov HaCohen-Kerner, Daniel Miller, and Yair Yigal. 2020. The influence of preprocessing on text classification using a bag-of-words representation, PloS one, vol. 15, p. e0232525.

Devamanyu Hazarika, Soujanya Poria, Sruthi Gorantla, Erik Cambria, Roger Zimmermann, and Rada Mihalcea. 2018. Cascade: Contextual sarcasm detection in online discussion forums. In COLING, pages 1837–1848. ACL.

Haibo He and Yunqian Ma (Eds.). 2013. Imbalanced learning: foundations, algorithms, and applications.

David W. Hosmer, Stanley Lemeshow, and Rodney X. Sturdivant. 2013. Applied logistic regression, Vol. 398, John Wiley & Sons.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. In IJCNLP, pages 757–762. ACL.

Aditya Joshi, Pushpak Bhattacharyya, and Mark J. Carman. 2017. Automatic sarcasm detection: A survey. ACM Computing Surveys (CSUR), 50(5), 1-22.

Jihen Karoui, Farah Banamara Zitoune, and Veronique ´ Moriceau. 2017. SOUKHRIA: Towards an irony detection system for Arabic in social media. Procedia Computer Science, 117:161–168. Arabic Computational Linguistics

Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. 2018. A large self-annotated corpus for sarcasm. In LREC. ELRA.

Bartosz Krawczyk. 2016. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, *5*(4), 221-232.

Diana G. Maynard and Mark A. Greenwood. 2014. Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. In Lrec 2014 proceedings. ELRA.

Bleau Moores and Vijay Mago. 2022. A Survey on Automated Sarcasm Detection on Twitter. arXiv preprint arXiv:2202.02516.

Sarang Shaikh, Sher Muhammad Daudpota, Ali Shariq Imran, and Zenun Kastrati. 2021. Towards improved classification accuracy on highly imbalanced text dataset using deep neural language models. Applied Sciences, 11(2), 869.

Samer Muthana Sarsam, Hosam Al-Samarraie, Ahmed Ibrahim Alzahrani, and Bianca Wright. 2020. Sarcasm detection using machine learning algorithms in Twitter: A systematic review. International Journal of Market Research, 62(5), 578-598.

Silviu Oprea and Walid Magdy, 2019. isarcasm: A dataset of intended sarcasm. arXiv preprint arXiv:1911.03123.

Silviu Oprea and Walid Magdy. 2020. The effect of sociocultural variables on sarcasm communication online. Proceedings of The 23rd ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW).

Tomáš Ptáček, Ivan Habernal, and Jun Hong. 2014. ˇ Sarcasm detection on czech and english twitter. In COLING, pages 213–223. ACL.

Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. 2015. Sarcasm detection on twitter: A behavioral modeling approach. In Proceedings of the eighth ACM international conference on web search and data mining WSDM, pages 97–106. ACM.

Byron C. Wallace, Choe, Do Kook Choe, and Eugene Charniak. 2015. Sparse, contextually informed models for irony detection: Exploiting user communities, entities and sentiment. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1035–1044. ACL.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcas as contrast between a positive sentiment and negative situation. In EMNLP, pages 704–714. ACL.

Sara Rosenthal, Preslav Nakov, Alan Ritter, and Veselin Stoyanova. 2014. Semeval-2014 task 9: Sentiment analysis in twitter. In Proceedings of the International Workshop on Semantic Evaluation (SemEval-2014), Dublin, Ireland, August.

Yan-yan Song and Ying Lu. 2015. Decision tree methods: applications for classification and prediction. Shanghai archives of psychiatry, 27(2), 130.

Tin Kam Ho. 1995. Random decision forests. Proceedings of 3rd international conference on document analysis and recognition. Vol. 1. IEEE.

Palak Verma, Neha Shukla, and A. P. Shukla. 2021. Techniques of Sarcasm Detection: A Review.

In 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE) (pp. 968-972). IEEE.