Hindawi
Complexity
Volume 2021, Article ID 2553199, 18 pages
https://doi.org/10.1155/2021/2553199

Research Article

Automated Prediction of Good Dictionary EXamples (GDEX): A Comprehensive Experiment with Distant Supervision, Machine Learning, and Word Embedding-Based Deep Learning Techniques

Muhammad Yaseen Khan,[1,2] **Abdul Qayoom**,[1] **Muhammad Suffian Nizami**,[3] **Muhammad Shoaib Siddiqui**,[4] **Shaukat Wasi**,[1] **and Syed Muhammad Khaliq-ur-Rahman Raazi**[1]

[1]*Center for Language Computing, Mohammad Ali Jinnah University, Karachi 75400, Pakistan*
[2]*Research and Development, Love For Data, Karachi 75600, Pakistan*
[3]*Dept. of Pure and Applied Science, University of Urbino 'Carlo Bo', Urbino 61029, Italy*
[4]*Islamic University of Madinah, Madinah 42351, Saudi Arabia*

Correspondence should be addressed to Muhammad Yaseen Khan; yaseenkhan.yzai@gmail.com

Received 25 June 2021; Accepted 14 August 2021; Published 13 September 2021

Academic Editor: Shahzad Sarfraz

Dictionaries not only are the source of getting meanings of the word but also serve the purpose of comprehending the context in which the words are used. For such purpose, we see a small sentence as an example for the very word in comprehensive book-dictionaries and more recently in online dictionaries. The lexicographers perform a very meticulous activity for the elicitation of Good Dictionary EXamples (GDEX)—a sentence that is best fit in a dictionary for the word's definition. The rules for the elicitation of GDEX are very strenuous and require a lot of time for committing the manual process. In this regard, this paper focuses on two major tasks, i.e., the development of labelled corpora for top 3K English words through the usage of distant supervision approach and devising a state-of-the-art artificial intelligence-based automated procedure for discriminating Good Dictionary EXamples from the bad ones. The proposed methodology involves a suite of five machine learning (ML) and five word embedding-based deep learning (DL) architectures. A thorough analysis of the results shows that GDEX elicitation can be done by both ML and DL models; however, DL-based models show a trivial improvement of 3.5% over the conventional ML models. We find that the random forests with parts-of-speech information and word2vec-based bidirectional LSTM are the most optimal ML and DL combinations for automated GDEX elicitation; on the test set, these models, respectively, secured a balanced accuracy of 73% and 77%.

1. Introduction

The comprehensive dictionary of any language provides the meaning of a word; at the same time, we find the correct usage of that word with an example of a sentence. Thus, when we can think of a *word*, a suite of multiple sentences can be set as examples to define it. All of these examples can be accurate w.r.t grammatical structure, the metaphor it delivers, and the context it is used into. In practice, with the corpus of these many (hundreds of thousands of) sentences against a single word, the lexicographers, under the activity of considering Good Dictionary EXamples (GDEX), try to elicit one particular sentence which best defines the very word on the qualitative grounds of being typical, informative, and highly readable [1, 2]. There are certain rules that the lexicographers have to take care of during the elicitation process. On these rules, for example, Kilgarriff et al. [2] have maintained that a good sentence is *one*—in an adequate length of 10–25 words, *two*—comprised of words that are in the top 17,000, *three*—consisting of target collocation in the main clause, *four*—not engaging pronouns and anaphors, *five*—provides a context, and *et cetera*. Overall, the activity is quite dawdling and sometimes it is converged into a compromising scenario when a good sentence is not good enough to be an example in the context of contemporary fashion. All of it eventually turns into a powerful need to substitute an automated GDEX elicitation process with artificial intelligence, which specifically deals with natural language processing (NLP) and natural language understanding (NLU).

The recent methods of automating such text classification tasks are based on supervised machine learning (ML) and Neural Network (NN) based deep learning (DL) techniques. These systems heavily rely on the prelabelled data, which mean, technically, a dataset that is labelled by humans. The accuracy of any such system is directly dependent on the size of data and quality of data labelling. However, recently, the researchers have produced abundant datasets for various classification tasks, but for the problem under study data is obscure, quite in deep relation to the fact that a lot of data is available over the Internet in the form of raw/unlabelled corpus; and if we aim to employ humans to do data labelling, a huge amount of time and labours efforts are required to complete it. In a parallel contrast, we have seen techniques such as distant supervision, which makes generalized assumptions for data labelling. For example instead of labelling a relation of Barack and Michelle marriage from the sentence "Michelle married Barack in 1992, and they have two daughters," we consider every sentence for marriage-relation where the terms Obama and Michelle appear [3]. Similarly, for sentiment analysis of product reviews, we can have binary star ratings supplied to it (such as the reviews with 3 or above stars out of 5 are positive, otherwise negative [4]).

Thus, for automation of such manual procedure of GDEX, in this paper, we have contributed to

(i) the development of a dataset using the distinct supervision technique for GDEX classification.

(ii) the application of supervised ML and DL algorithms to predict whether, for a given word, a sentence in running English text is good or not.

(iii) the comparative analysis on the robustness and trade-off between ML and DL approaches.

(iv) the competitive analysis between manual GDEX elicitation routines and automated GDEX classification.

However, it does not mean that the proposed methodology explicitly examines the syntax and other linguistic elements of good writing, nor does it deal with the inference of polarity (under the computational study for effect) in the given text, which, in general convention, refers to the task of sentiment analysis. Instead, as prefatory research, it aims to verify whether a discriminative classifier can be sought for categorizing English sentences as either of the binary classes good and bad through the supervised ML algorithms.

This paper is systematically divided into 5 subsequent sections, where the related work done for the same problem is given in Section 2. Section 3 provides details on the material and methods: data source, data labelling strategy, and approaches followed by maintaining information on ML and DL methods. The insights into the results, critique, and comparative and completive analyses on the results are presented in Section 4. The conclusion of the paper and future work are given at the end.

2. Literature Review

On the problem under study, there are many significant methodologies proposed by researchers; however, we maintain that, in comparison to other classification tasks in NLP, the amount of work for GDEX classification is small.

Pilán et al. [5] made the most relevant work for GDEX classification; they have developed a system to evaluate either the sentence suitability for dictionary examples or good examples for teaching purposes. They argue that a good example should be typical, informative, and intelligible and should be easily readable for the learners. The two techniques based on natural language processing and machine learning were used for sentence selection. The content has been taken from Swedish novels, newspapers, and blogs for applying both techniques. From this work, 70% of the total sentences were suitable for understanding by students and teachers. Srdanović and Kosem [6] presented GDEX classification for the Japanese language; it was designed mainly for the lexicography of the Japanese language and learning purposes. In this research, a randomly extracted list of lemmas was used for evaluating GDEX configurations.

Kilgarriff et al. [2] presented some rules and boundaries for a good sentence; according to the study, the sentence should hold the following characteristics (or comply with the following rules):

(i) (Rule#1) A sentence consisting of 10 to 15 words will be preferred.

(ii) (Rule#2) A sentence will be penalized when it does not lie among 17,000 commonest words in a language.

(iii) (Rule#3) A sentence containing pronouns and anaphors will be penalized.

(iv) (Rule#4) Target collocation should be in the main clause.

(v) (Rule#5) A sentence should start with a capital letter and end with a full stop, exclamation sign, or question mark.

Moreover, for a GDEX, Kilgarriff et al. [2] eulogize the first two characteristics/features (sentence length and word frequency) should be given the highest weight as compared to other features. According to Kosem et al. [7], the most important characteristics of a GDEX are authenticity, typicality, informativeness, and intelligibility. The developers of Good Dictionary EXamples system and their configurations are often lexicographers and lack programming skills in many cases.

Geyken et al. [8] show that GDEX work can be extended through ML techniques for mapping example sentences to dictionary sense. They performed the computations of all collocations sets and then maximum entropy [9] was used for learning the correct mapping between corpus sentences and their correct dictionary sense. Ljubešić and Peronja [10] presented another ML approach to extract GDEX. The dataset used in their experiment contains several examples of sentences with annotations of four classes/levels (i.e., very bad, bad, good, and very good). They used the Random Forest regressor algorithm [11] and secured an average precision of 90%.

Stanković et al. [12] gave a similar work for the selection of GDEX for Serbian and it was used for the development of preliminary components of the model. Their approach analyses the lexical and syntactic aspects of a corpus consisting of five digitized volumes of examples from the Serbian Academy of Sciences and Arts (SASA) dictionary. They compared the feature distribution of examples from their corpora with the feature distribution of sentence samples extracted from corpora comprising various other texts. This way, selected candidate 140 examples were represented as feature vectors, and supervised machine learning classifiers were used for standard and nonstandard Serbian sentences.

Koppel [13] presented work for GDEX classification in the Estonian language. The group used the web corpus of etTenTen13; in their approach, they focus on the sentence length, word length, the number of subordinate clauses, and keyword position. In another similar study, Uprety and Shakya [14] conducted a test to analyse the effectiveness of context clue sentences among Nepalese students. Their study results showed that context clue sentences were more useful in learning vocabulary words than GDEX sentences. Based on their research results they concluded and recommended that context clue sentences should be included in the Good Dictionary EXamples to help the new learners.

3. Materials and Methods

This section is divided into three subsections; each one is dealing with the focused methodology such as data gathering and labelling (Subsection 3.1), preprocessing and feature selection (Subsection 3.2), and an overview about experiment setup employing the suit of predictive algorithms for machine learning (Subsection 3.3).

3.1. Data Source and Data Labelling. We prepared our dataset in the fashion of distant supervision. Using BeautifulSoup (https://www.crummy.com/software/BeautifulSoup/bs4/doc/) we scraped sentences from the website sentence.yourdictionary.com (YD.com). The scraping is made for the top 3K English words listed by Oxford Learner's Dictionaries (https://www.oxfordlearnersdictionaries.com/wordlist (accessed May 20, 2021)). On average we have got ≈250 sentences for a single word and more than 785K English sentences in total. Furthermore, the website not only provides the example sentences, but it also presents the count of thumbs-up and thumbs-down for every sentence against the very word. Hence we maintain the corpus in dictionary structure where, for every word as a key, a list of tuples is retained. To mean it mathematically, consider equation (1) below:

$$C \leftarrow \left\{ \begin{array}{l} w_0 \longrightarrow [\, \langle S_0^{w_0}, U_0, D_0 \rangle,\ \langle S_1^{w_0}, U_1, D_1 \rangle,\ \cdots], \\ w_1 \longrightarrow [\, \langle S_0^{w_1}, U_0, D_0 \rangle,\ \langle S_1^{w_1}, U_1, D_1 \rangle,\ \cdots], \\ \qquad\qquad\qquad \vdots \\ w_n \longrightarrow [\, \langle S_0^{w_n}, U_0, D_0 \rangle,\ \langle S_1^{w_n}, U_1, D_1 \rangle,\ \cdots], \end{array} \right\} \tag{1}$$

where C is a dictionary with key-value pairs such as word w being the key, against whom a list of tuples is retained; further, the contents of the tuple shows the example sentence S_i^w along with its thumbs-up votes (U_i) and thumbs-down votes (D_i); the subscript i indicates the index of sentence respectively. The target label of a sentence, i.e., *good* or *bad* (or 1 and -1 in respective order), is determined by the count of thumbs-up and thumbs-down votes. In further analysis, we notice that YD.com holds different votes for the same sentence if the very sentence is referenced as an example to the different words. Hence, C is of no use if there exist redundant sentences with different votes. To restructure the dataset we extract a set of distinct sentences S^* from C as per the following equation:

$$S^* \leftarrow \left\{ C_{[w][i][0]} | w \in C; \quad \forall_i \in I = \left\{0, 1, \ldots, \left\| C_{[w]} \right\| - 1 \right\} \right\}. \tag{2}$$

Further, we prepared different datasets—corresponding to the pooling function $\Psi(\cdot)$—having sentences and their labels in the form of tuples with the manner shown in the following equation:

$$\widehat{C} \leftarrow \{ \ (s_0, \lambda_0),\ \ (s_1, \lambda_1),\ \cdots\ \ (s_n, \lambda_n), \} \tag{3}$$

where $s_. \in S^*$ and λ_i is the label of respective i^{th} sentence in $\widehat{C}$ and determined on the criteria under function Φ given in the following equation:

$$\lambda_i \leftarrow \Phi(a_1, a_2) \leftarrow \begin{cases} \text{none}, & \text{if } a_1 = a_2 = 0, \\ \text{good}, & \text{if } a_1 \geq a_2, \\ \text{bad}, & \text{otherwise}. \end{cases} \tag{4}$$

In equation (4), a_j is a real number yielded through a pooling function $\Psi_p(\cdot)$ (explained later in the following text). In $\Phi(a_1, a_2)$; subscript j for a_j specifically indicates the incidences for $U_.$ and $D_.$; hence, $j \in \{1, 2\}$, to mean thumbs-up votes $a_j = 1$ and thumbs-down votes $a_j = 2$. Lastly, the value of a_j is calculated as per the following equation:

$$a_j \leftarrow \Psi_f \left(\left\{ C_{[w][i][j]} | s \in S^* \wedge w \in C \wedge s = C_{[w][i][0]}; \forall i \in I \right\} \right), \tag{5}$$

where p is a final score calculating function, $f \in \{\max, \text{average}, \text{sum}\}$. The index set I in the equation above has already been defined in equation (2). Thus, we utilized these votes as the crowdsourced labelling and adjudged a sentence to be good if the total thumbs-up votes are equal to or greater than thumbs-down votes (see equation (4)).

Table 1 gives the statistical information on the labelled dataset that is employed in this experiment. The dataset for every scoring function is balanced, i.e., each class contains 20K records (which alternatively means 40K sentences, in total, are used in the experiments.) One key observation we can get from the table is the average sentence length of good examples is approximately half of its counterclass. It further asserts that the distinct supervision (or nearly crowdsourced data) appeared to have aligned with rule#1 (i.e., already stated in Subsection 2.2).

3.2. Machine Learning-Based Classification: Feature Enhancement, Transformation, and Algorithms. At the beginning of this section, the authors would like to maintain a summarized idea of experiments conducted for the GDEX classification based on the conventional ML algorithms; in the same context, the following itemized text provides a brief commentary on the components depicted in Figure 1.

(i) We experimented with two different approaches for the feature enhancement, such as the following:

   (1) Bag of Word (BoW).
   (2) Usage of Part of Speech (PoS) tags alongside the words.

(ii) Besides the above two approaches, we set two feature transformation (or vectorization) techniques for the sentences in the dataset, such as the following:

   (1) Word frequency-based count vectorization.
   (2) Term Frequency-Inverse Document Frequency ($TF * IDF$) features normalization-based vectorization.

(iii) The combination of these feature enhancement approaches and feature vectorization techniques are evaluated under the five conventional ML algorithms in the 10 randomly generated training and testing subsets under the Monte Carlo method. The ML algorithms used in this paper are enumerated below:

   (1) k-nearest neighbours (k-NN).
   (2) Naïve Bayes (NB)/Gaussian Naïve Bayes (GNB).
   (3) Random Forest Trees (RFT).
   (4) Linear Support Vector Machines (linear-SVM).
   (5) SVM with radial basis function kernel (rbf-SVM).

Thus, the total number of experiments set for ML-based GDEX classification is 60, i.e., 2 (feature enhancement approaches) $\times$ 2 (feature vectorization) $\times$ 5 (ML algorithms) $\times$ 3 (datasets yield from the three different final scoring functions) = 60. The details of each of these components are provided in the subsequent subsections.

3.2.1. Feature Enhancement Approaches. The BoW approach is considered a very basic approach in any task in NLP [15]. It consists of tokenization of a running text/document and submission of tokens for further process. However, we can think that these sequences of words are of more importance and become meaningful and informative when they are analysed with the corresponding PoS tags. Thus, hundreds of papers in the domain of NLP and NLU utilized such information of words' PoS alongside words in their capacities [16, 17]. In the same regard, we can anticipate the words in addition to respective PoS tag information (BoW + PoS) will attain more robustness in the predictive ML model with two significant hypotheses:

(i) BoW + PoS creates highly discriminative features for classifying a GDEX.
(ii) Forbye the previous point, BoW + PoS embodies a writing pattern that exists for a comparatively longer sequence in n-grams—we surmise that it may engage better syntactic and semantic attributes.

On a technical note, we have used Natural Language ToolKit (NLTK) based word tokenizer (https://www.nltk.org/api/nltk.tokenize.html; there are many tokenizers provided in the module; function, which is precisely used in this paper, is, namely, `word_tokenize`) for the sentence tokenization; followed by it, the PoS tagging is also done with NLTK-based PoS tagging module (https://www.nltk.org/api/nltk.tag.html#module-nltk.tag). We concatenated the word and its respective PoS tag with an underscore, as it is shown for a single sentence in Table 2; however, the information on the tag-set can be accessed in the online documentation of NLTK (https://www.nltk.org/book/ch05.html).

3.2.2. Feature Vectorization Techniques. ML algorithms are not supposed to work directly on the running texts. Since there are thousands of terms in the vocabulary and a few of them are appearing in a sentence, we are required to transform every sentence through a specific mechanism that applies to all of the sentences in the dataset and is hence workable for the ML algorithms. Typically, the sentence transformation mechanism takes a sentence and projects it into a high-dimensional vector space [15]. The final structure of the dataset will be a matrix. It contains the number of rows equal to the number of sentences and the number of columns as per the size of vocabulary (or in other words, the dimensionality of a single vector is equal to the size of vocabulary). Thus, we can think of the values on the dimensions, corresponding to the words present in the sentence and carry nonzero numeric values; otherwise, they are zero (in the case of nonsparsity). The very matrix can be sparse by ignoring the indexing of words/dimensions that are not present in the sentence and retaining the records for the words that appear in the sentence.

Count vectorization, which is the first vectorization technique used in this paper, involves sentence vectorization by keeping the count of words that appear in the sentence and zero at the remaining dimensions. Figure 2 illustrates the count vectorization process, in which the first step includes generating a dictionary for word-indices, followed by utilizing very dictionary for the transformation of sentences in vector space.

Table 1: Class distribution and statistical insights into the labelled dataset.

Class	Final scoring function Ψ_f	Count of tokens	Count of distinct terms	Average words in sentence
Good (1)		302,389	22,200	$15.12 \approx 15$
Bad (−1)	f = avg	536,604	40,232	$26.83 \approx 27$
Total		838,993	47,028	$20.97 \approx 21$
Good (1)		287,794	26,228	$14.39 \approx 14$
Bad (−1)	f = sum	627,952	57,686	$31.4 \approx 31$
Total		915,746	64,840	$22.89 \approx 23$
Good (1)		279,894	25,738	$13.99 \approx 14$
Bad (−1)	f = max	610,676	57,604	$30.53 \approx 31$
Total		890,570	64,728	$22.26 \approx 22$

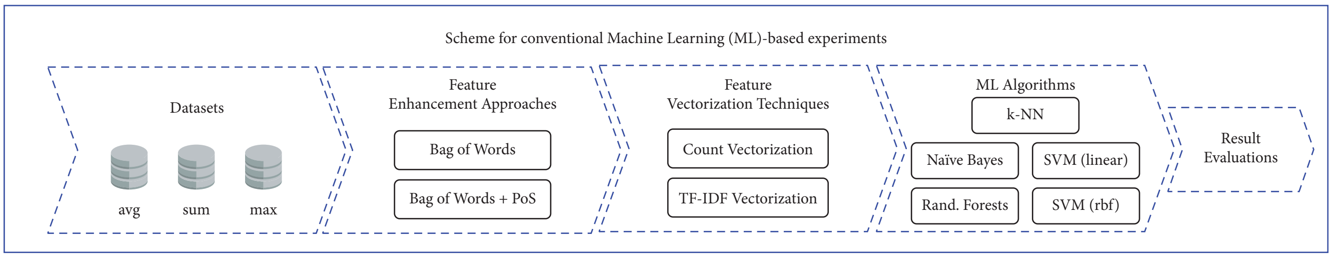


Figure 1: Overall scheme of experiments designed for conventional machine learning.

Table 2: Example for the simple word tokenization and PoS tag induction.

Example	Laura always remained an object of curious study.
BoW features	[‘Laura’, ‘always’, ‘remained’, ‘an’, ‘object’, ‘of’, ‘curious’, ‘study’, ‘.‘]
BoW + PoS features	[‘Laura_NNP’, ‘always_RB’, ‘remained_VBD’, ‘an_DT’, ‘object_NN’, ‘of_IN’, ‘curious_JJ’, ‘study_NN’, ‘._.‘]

We may think of the cases where the most frequent words (i.e., a, an, the, of, to, *et cetera*, known as stop words) dominate in a sentence—diminishing the impact on least frequent words—hence, resulting in a larger value on their respective dimensions. In this regard, the TF ∗ IDF approach sets a trade-off between the high-frequent and less-frequent words [15, 18]. It works by calculating a product of term frequency relative to a document (TF) and inverse document frequency of the very term in the corpus (IDF). To mean the TF and IDF mathematically, consider the following equations; moreover, Figure 3 uses these formulae to illustrate TF ∗ IDF calculations.

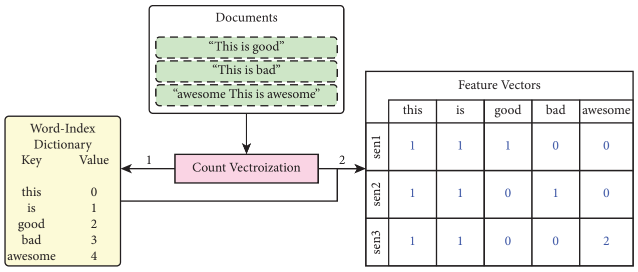
$$\begin{aligned}\mathrm{TF}(t,d) &= \frac{\text{frequency of term } t \text{ in document } d}{\text{total number of terms in the document}} \\ &= \frac{f_{t,d}}{\sum_{t'\in d} f_{t',d}}, \\ \mathrm{IDF}(t,D) &= \log\frac{\text{total number of documents in the dataset}}{\text{number of documents containing the term } t} \\ &= \log\frac{|D|}{\left|\{d\in D\colon t'\in d\}\right|}.\end{aligned} \tag{6}$$

On a technical note, we have used the n-gram range [1, 3] in sklearn, which assumes the formation of unigrams, bigrams, and trigrams in the input string. Alongside it, we kept the same tokenization function for both of the vectorization processes, which has already been discussed in the previous subsection.

3.2.3. Machine Learning Algorithms. K-Nearest Neighbours is among the instance-based lazy learning techniques in conventional ML algorithms [19, 20]. Functionally, it computes the distance between the target document vector and all of the document vectors, followed by selecting k documents where the distance is minimum. In last, it decides the class for the target document through voting in the k-nearest neighbour vectors. The number of neighbours set for this work is five (i.e., $k = 5$). Furthermore, we like to maintain that there are many measures for computing distances between documents, and the one we have employed in this paper is cosine similarity. Since similarity is inversely proportional to the distance, with the case of similarity, the k-NN algorithm will perform voting on the k documents with the maximum similarity. The value of the cosine similarity ranges in [0, 1], where the similarity score 0 indicates no similarity whereas 1 indicates absolute similarity. The cosine similarity between two document vectors (A and B) is calculated through the following equation [15]:

$$\mathrm{Similarity}(A,B) = \cos(\theta) = \frac{A\cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}. \tag{7}$$

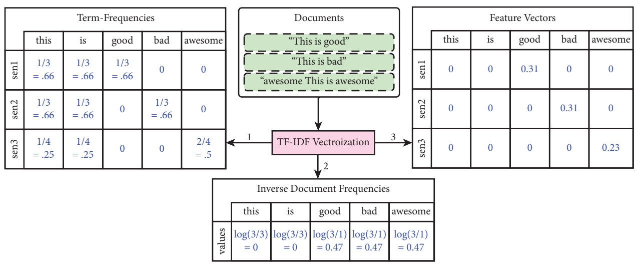
Naïve Bayes is a conventional ML algorithm for classification tasks [4, 15]. It classifies the sentences by exploiting conditional probability using Bayes’ theorem; however, the basic assumptions naïve Bayes holds are of the conditional independence between the features. The basic calculation



Key	Value
this	0
is	1
good	2
bad	3
awesome	4

	this	is	good	bad	awesome
sen1	1	1	1	0	0
sen2	1	1	0	1	0
sen3	1	1	0	0	2

Figure 2: Illustration of count vectorization.



	this	is	good	bad	awesome
sen1	1/3 = .66	1/3 = .66	1/3 = .66	0	0
sen2	1/3 = .66	1/3 = .66	0	1/3 = .66	0
sen3	1/4 = .25	1/4 = .25	0	0	2/4 = .5

	this	is	good	bad	awesome
values	log(3/3) = 0	log(3/3) = 0	log(3/1) = 0.47	log(3/1) = 0.47	log(3/1) = 0.47

	this	is	good	bad	awesome
sen1	0	0	0.31	0	0
sen2	0	0	0	0.31	0
sen3	0	0	0	0	0.23

Figure 3: Illustration of TF $*$ IDF vectorization.

done by the naïve Bayes for classifying a sentence (X) is given in the following equation:

$$f\left(C_k|X\right) = \frac{p\left(X|C_k\right)p\left(C_k\right)}{p(X)}. \tag{8}$$

Equation (7) is expanded w.r.t the individual features $\left(X = \left\{x_0, x_1, \ldots, x_n\right\}\right)$; see equation (9) below:

$$\begin{aligned} f\left(C_k|x_0, x_1, \ldots, x_n\right) &= p\left(C_k\right)p\left(x_0|C_k\right)p\left(x_1|C_k\right)\cdots p\left(x_n|C_k\right) \\ &\approx p\left(C_k\right)\prod_{i=0}^{n} p\left(x_i|C_k\right). \end{aligned} \tag{9}$$

However, when the documents are normalized and transformed through the TF $*$ IDF vectorization, the values for features are no longer discrete. Thus, for the continuous features, we cannot employ the above conventional naïve Bayes algorithm. Instead, we have to use its variant that uses Gaussian distribution (and hence, known as Gaussian naïve Bayes) [21, 22]; the substitution of $\left(x_0|C_k\right)$ in the Gaussian naïve Bayes is defined in the following equation:

$$p\left(x = v|C_k\right) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\left(\left(v-\mu_k\right)^2/2\sigma_k^2\right)}. \tag{10}$$

Finally, the target class $\hat{y}$ (by either of conventional naïve Bayes or Gaussian naïve Bayes) is elicited where (.) is maximum; to mean it mathematically, see equation (11), where K is the set of classes:

$$\hat{y} = \underset{k\in\{1,\ldots,K\}}{\arg\max}\, p\left(C_k\right)\prod_{i=1}^{n} p\left(x_i|C_k\right). \tag{11}$$

Random Forest is an ensemble approach in ML classification algorithms, which is based on Decision Trees (DT) [23]. Instead of relying on a single decision tree, the basic aim is to draw multiple decision trees from the bootstrapped-random samples of training data. The testing data will be predicted on each of the DT, followed by eliciting the final label through voting [11]. Thus, we can think of RF overcoming the issue of overfitting through ensemble technique. Figure 4 shows how the RF classifier works and outputs a final class from all of the DTs. In the experiment, we have used 200 trees (or DT estimators) for building a forest.

Support Vector Machine is one of the widely employed classifiers in conventional ML algorithms [24]. It is well suited for the classification of complex, imbalanced ones but should be small or medium-sized datasets. The SVM aims to draw a hyperplane in an n-dimensional vector space, such that the hyperplane separates data points into two distinct partitions of data, representing the respective classes [25]. The SVM can be used for linear or nonlinear classification. However, the basic SVM, which fits a hyperplane, is conventionally known as linear-SVM [25, 26]. Equation (12) gives the mathematical semantics for understanding linear-SVM.

$$\hat{y}_i = \begin{cases} 1, & \text{if } w^T \cdot X_i - b \geq 1, \\ -1, & \text{if } w^T \cdot X_i - b < 1. \end{cases} \tag{12}$$

In this work, we have used linear-SVM and radial basis function (rbf) SVM (through kernel trick). The basic objective rbf-SVM sets are to fit a circular boundary margin for nonlinear datasets. The illustration in Figure 5(a) shows the linear-SVM, and in contrast, Figure 5(b) shows the situation where a hyperplane is not suitable for separating datasets into two distinct parts; and instead, this can be achievable

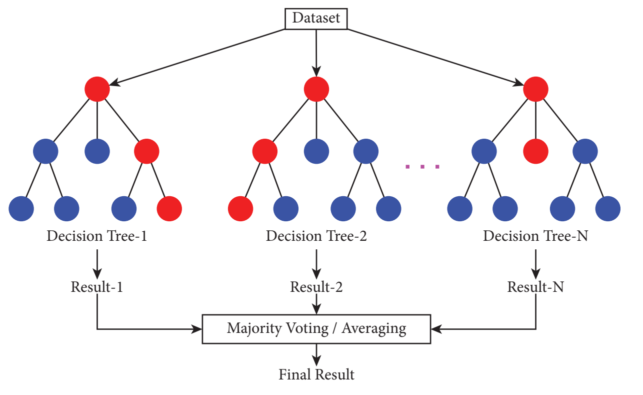


FIGURE 4: Illustration of random forest trees.

only with the rbf kernel trick. The red and blue dots are representing separate classes. The solid line in grey colour is the decision boundary; dots coinciding with the dashed line are termed as support vectors.

3.3. Feature Engineering for Deep Learning-Based Classification. In this section, a discussion on the DL models and input data encoding schemes are given in detail. Likewise, in an earlier summary methodology involving ML in Subsection 3.3, the authors would like to maintain a brief commentary on DL-based models; Figure 6 shows the overall scheme for these experiments.

(i) Since we empirically found, in the suite of ML-based algorithms, the most optimal result was secured with the dataset based on a final scoring function $\Psi_{f=sum}$, all of the DL-based experiments are performed only on the aforesaid dataset.

(ii) Since the NN essentially requires input data to be encoded in a numeric form, for doing the needful, we used 3 different data encoding approaches, which are as follows:

 (1) One-hot encoding.
 (2) Global Vectors (GloVe) based embedded encoding.
 (3) word2vec-based embedded encoding.

(iii) A combination of these data encoding approaches was made with the following 5 DL algorithms/networks:

 (1) Recurrent Neural Network (RNN).
 (2) Gated Recurrent Unit (GRU).
 (3) Long-Short Term Memory (LSTM).
 (4) Bidirectional GRU (Bi-GRU).
 (5) Bidirectional LSTM (Bi-LSTM).

(iv) We did not perform any feature engineering (for example, extraction and usage of PoS tags) in the DL-based experiments, because, congenitally, the NNs are taken as to learn and accommodate and intrinsic features present in the data.

(v) All of the DNNs used in the methodology are programmed with the Python-based Keras (https://keras.io) library, which uses Tensor Flow 2 (https://www.tensorflow.org) at the backend for processing. Moreover, the experiments are run at Google Colaboratory (https://colab.research.google.com) on the GPU-accelerated runtime.

Thus, the total number of experiments done with DL-based methods is 15, i.e., 3 (data encoding approaches) × 5 (DNNs) = 15. The detail of these components is given in the subsequent subsections.

3.3.1. Data Encoding Approaches. The NNs need data to be in numeric form for which we have got many transformation or encoding approaches. One-hot encoding is one of the techniques among them, which generates a single vector against every word in a sentence, such that the index corresponding to the very word is 1 and the rest of all incidences are 0. Thus, we can see a sparse matrix-like structure (or a list of four one-hot vectors) for the sentence "This is a cat" as is illustrated in Figure 7(a). Each row of the yellow block is a vector where there exists only a single entry of 1, indicating the presence of the very word in the vector. Hence, with this technique, we can think that input data is sparse and exists in a very high-dimensional space.

In contrast, the second approach for data encoding is based on NN inspired word embeddings and statistical means, which are dense and adjustable to any of the n-dimensional spaces, provided that $n > 0$; Figure 7(b) illustrates the example of word embedding where each row in blue colour is a dense representation of the word in 4-D space. The word embeddings render meanings to Firth's philosophy "You shall know a word by the company it keeps!" [27] through realizing the capability of retaining the context of words, such that every word will exist alongside the similar words (using GloVe, the examples of the nearest words for the word "king" are "kings," "queen," "monarch," *et cetera*; retrieved through online tool available at http://bionlp-www.utu.fi/wv_demo) in the n-dimensional space of word embeddings. In this work, we have employed two different word embeddings, namely, word2vec [28] and GloVe [29], developed by Google and Stanford, respectively. In addition to this, the word2vec employs continuous BoW in NN for learning the prediction of the current word (given the input of context of words) and skip-grams for learning the similar words (given a source/input word), whereas the GloVe utilizes matrix factorization techniques such as Latent Semantic Analysis (LSA) [30] on word-word context matrix for generating word vector representations. On a technical note, the representation used in this work is based on 300 dimensions (these vectors can be accessed at http://vectors.nlpl.eu/repository).

3.3.2. Deep Neural Networks. The NNs are the computational system of connected units that loosely simulate the working of biological neurons in the brain of living beings. The story of ideas and advancements made in the file of NN is historic. (The earlier NNs are devised by McCulloch and Walter [45], in 1943, for artificially simulating the working of a biological neuron

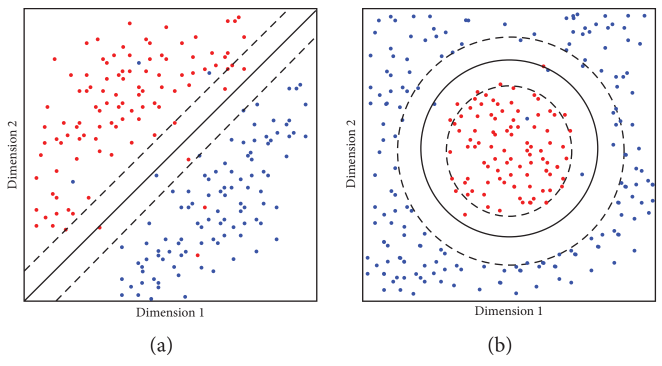


Figure 5: Illustration of linearity and nonlinearity in data. (a) The linear-SVM corresponds to fitting a hyperplane when the data reflects its shape as depicted in the subfigure *a*, (b) whereas the subfigure *b* reflects the situation optimal for rbf-based SVM.

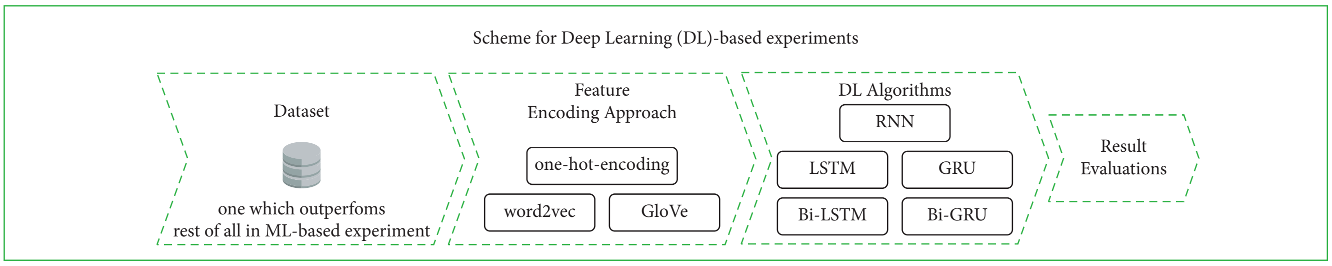

Scheme for Deep Learning (DL)-based experiments
Dataset
one which outperfoms rest of all in ML-based experiment
Feature Encoding Approach
one-hot-encoding
word2vec
GloVe
DL Algorithms
RNN
LSTM
GRU
Bi-LSTM
Bi-GRU
Result Evaluations

Figure 6: Overall scheme of experiments designed for deep learning.

[39, 46]. This earlier work is rendered with computational means known as "Calculators" and "Perceptron," respectively, in 1954 by Farley and Clark [47] and in 1958 by Rosenblatt [48]; however these works were limited to present the working of single neuron [39, 44]. Upgrading NN with several layers (thus, called DNN) was made in 1965 by Ivakhnenko and Lapa [49]. In 1975, Werbos [50] presented that the backpropagation technique can be used for new weights learning for the training of multilayer networks [46]; the further work done by Rumelhart et al. [51] showed that the backpropagation techniques learn interesting features for text processing.) However, the authors would like to maintain a brief introduction to the basic working of these connected units or a NN (which is also illustrated in Figure 8), where inputs (or signals) received at the input layer are analysed and transmitted to further neurons to which they connected. We know the input should be a numeric value (for which we have maintained information in the previous section); thus, the input $(X = \{x_0, x_1, \ldots, x_n\})$ received at the units of the hidden layer and the respective weights $(W = \{w_0, w_1, \ldots, w_n\})$ that are correspondingly associated with the edges are taken for the dot product $(\sum = X \cdot W)$— creating a linear output. In the next step, bias (b) is added to this linear output $(z = X \cdot W + b)$, and the result is converted into nonlinearity through passing it to a nonlinear activation function, that is, in our case, tan *h* function, which is given in equation (13).

$$\tanh(z) = f_h(z) = \frac{e^{2z} - 1}{e^{2z} + 1}. \tag{13}$$

Similarly, the output of hidden neurons is transmitted to the final output neuron that takes a step function to compute the class of given input data. The step function, which is used in this paper is sigmoid that returns a number in the range of [0, 1], where we consider the prediction is relating to the positive class if the value is above 0.5; otherwise it belongs to the negative class. The sigmoid function is given in the following equation:

$$\hat{y}_i = \sigma(z) = \frac{1}{1 + e^{-z}}. \tag{14}$$

The backpropagation technique is used to update weights considering the prediction errors that occurred during the training. In this context, DNN typically divides training set into multiple batches; thus, with one batch it calculates the error followed by updating the new values for the weights. Executing the same process on each batch will mark one run, which is technically called an epoch.

In this paper, we have used three types of NN that were specifically developed for text (or generally known for sequence) processing. The RNN [31] is one of the first DNNs that attempted to involve input history in the sequential data such that the process of RNN moves onwards with subsequent inputs alongside incorporating the result (of the hidden state) of the previous input units.

W.r.t Figure 9, the RNN works for every timestamp *t*, and the hidden state $a^{\langle t \rangle}$ and the output $\hat{y}^{\langle t \rangle}$ are expressed as per equations (15) and (16).

$$a^{\langle t \rangle} = g_1\left(W_{aa} a^{\langle t-1 \rangle} + W_{ax} x^{\langle t \rangle} + b_a\right), \tag{15}$$

$$\hat{y}^{\langle t \rangle} = g_2\left(W_{ya} a^{\langle t \rangle} + b_y\right), \tag{16}$$

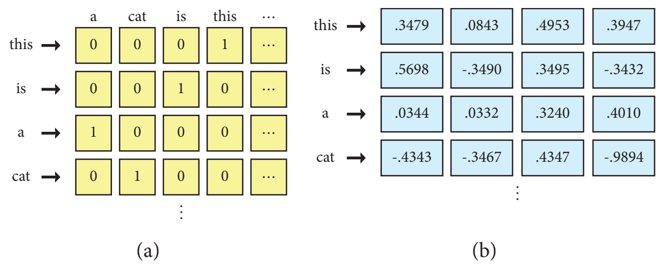


Figure 7: Illustration of the one-hot encoding and word embeddings. (a) One-hot encoding, and (b) word embedding.

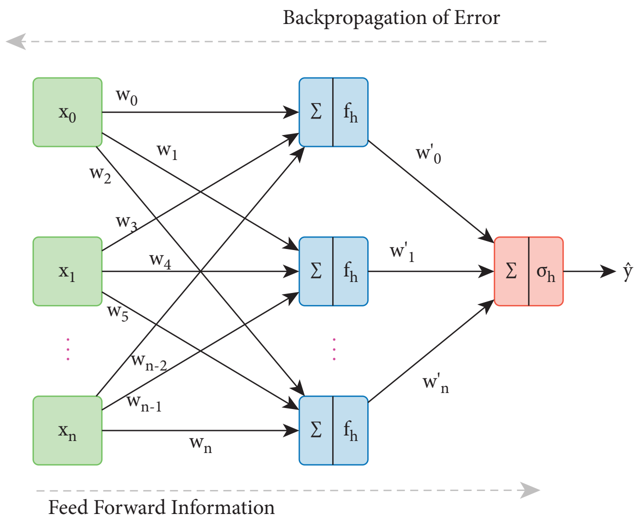


Figure 8: Illustration of the artificial neural network.

where $W_{aa}, W_{ax}, W_{ya}, b_a$, and b_y are the coefficients, and f is the activation function; comprehension of these coefficients and the internal structure of the blue box (illustrated in Figure 9) are given in Figure 10.

The RNNs though were developed to retain memory but instead, they failed on doing it for the longer sequences. Alternatively, Hochreiter and Schmidhuber [33] presented another RNN-based architecture, namely, LSTM, which served better for the problem of input retaining. The LSTM introduced the concept of the gate for remembering the inputs; however, later an upgraded form of LSTM is presented by Gers et al. [34], which added forget gate in the architecture; further, with the induction of forgetting gate LSTM became capable of resetting its state [35]. The LSTM though is the wonderful RNN architecture but it takes more memory and processing time [36, 37]. Cho et al. [38] introduced GRU, which is alike LSTM but contains fewer parameters. Traditional RNNs suffer the vanishing gradient problem, which is handled at the optimal level in LSTM and GRU [32, 33, 39]. The bidirectional LSTM and GRU are the variant of vanilla LSTM and GRU, which are capable of making the DNN process string in forward and backward directions [39]. In Table 3 the summary of gates used in LSTM and GRU is presented, in addition to which we can see their usage in the illustrations of LSTM and GRU in Table 4, where $\odot$ shows elementwise multiplication between two vectors.

The networks we have employed in this paper have the same input and output layer.

However, the hidden layer varies w.r.t the architecture. This DNNs are programmed with Keras using the sequential model. Information on the layers hyperparameters used in this work is given in Tables 5 and 6.

4. Results and Discussion

In this section, we presented a thorough discussion on the evaluation and comparisons of the ML and DL models. However, before proceeding any further, it should be in the knowledge that the evaluation is done on a validation set which is extracted from the labelled corpus with Pareto principle or 80/20 rule [15, 40]. These details are maintained in separate subsequent subsections.

4.1. Evaluation Criteria and Metrics. The classification task in a supervised learning domain is often evaluated through the confusion matrix (CM), which statistically presents the number of correct and incorrect predictions w.r.t. the actual labels in the validation set. A sample CM is given in Table 7, where TNs are the *true negatives*, which logically means the number of actually negative documents and predicted negative as well; TP (*true positives*) will mean exactly the opposite to TN (i.e., consider a positive class in place of negative). The FP is the *false positives*, which logically means the number of documents that are actually negative but misclassified as positives; FNs (*false negatives*) are the exact opposite of FP, such that they were the misclassified documents that were actually negative but falsely predicted as negative.

We can drive several evaluation statistics for assessing the quality of the Predictive System (PS) using the CM. The statistics and their derivations used for the assessment of ML and DL models in this work are defined in Table 8. Moreover, for the ideal PS, we expect to have the highest value on the left diagonal of every individual CM, whereas, at the right diagonal of the matrix, we expect the least value.

For the evaluation of performance in this paper, we consider R and BA are of more importance. The R is critical because we consider losing or misclassifying a positive document into another category is perilous—as we have got little data for the GDEX classification in comparison to the colossal dataset—thus, we will consider an ML or DL model with an optimal where the R is higher. In a similar context, this will not mean the small value of S; hence, the BA is the

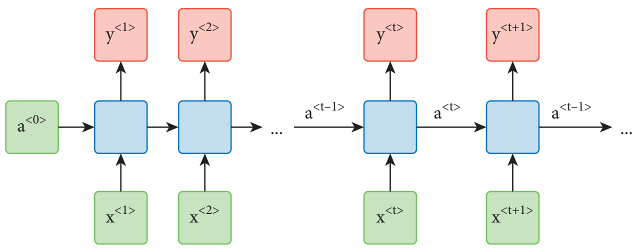


FIGURE 9: Illustration of the RNN. Image courtesy [32].

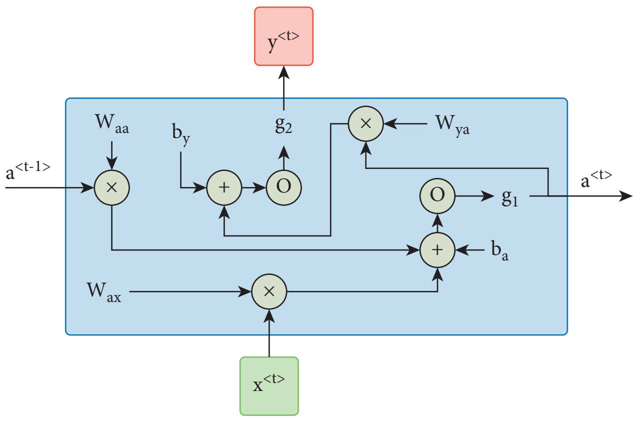


FIGURE 10: Depiction of the internal architecture of RNN. Image courtesy [32].

ultimate choice for the fair evaluation, which encompasses both of the statistics relating to TP and TN.

4.2. Analysis of ML Models and Results. The quantified statistics of all evaluation metrics are given (respectively, for the final scoring functions, i.e., $\Psi_{f=\text{avg}}, \Psi_{f=\text{max}}, \Psi_{f=\text{sum}}$) in Tables 9–11 . The overall result of the ML-based models is positive. We can see an obvious insight into the better performance of all ML models (in all respective datasets corresponding to the final scoring functions) that are vectorized through the TF ∗ IDF approach. On the collective ground, the dataset created with $\Psi_{f=\text{sum}}$, parse, indicates the most optimal method for the dataset creation through distant supervision. In contrast, the results with the $\Psi_{f=\text{avg}}$ show the least significance for making the discriminant dataset for predictive modelling; hence, we can maintain that the distant supervision cannot be used with the averaging methods for data curation in supervised learning tasks.

Since the dataset with $\Psi_{f=\text{sum}}$ shows better results, we will consider it (considering Table 11) for the discussion in the remaining text. Coming towards the evaluation of feature enhancement technique, we see the BoW + PoS tags show better results in comparison to the only BoW approach. However, a drastic change in accuracy of *k*-NN (i.e., w.r.t $\Psi_{f=\text{sum}}$, improvement of +12% with count vectorization and +2% with the TF ∗ IDF vectorization) is seen when the PoS information is inducted alongside the simple words. However, in comparison to the count vectorization technique, we maintain that the improvement with the additional PoS information is slightly more visible in the TF ∗ IDF vectorization technique.

The most optimal ML algorithm and combination found with maximum accuracy of 77.3% are rbf-SVM + TBP. (TBP will be the acronym for the combination of TF ∗ IDF vectorization + BoW + PoS tags features. Similarly, CBP will be combination of count vectorization + BoW + PoS tags features. TB will stand for the combination of TF ∗ IDF + BoW features; and *CB* will be count vectorization + BoW features.) Ignoring the trivial difference of linear-SVM with its other variant, we can consider RFT + *TBP* secures the second position by attaining accuracy of 76.8%. For BA, *k*-NN + *TBP* is found the best combination with a 75.5% score, followed by RFT + *TB* with securing a 73.9% score. Besides accuracy and balanced accuracy, the highest recall (i.e., 75.4%) is seen in a dataset with $\Psi_{f=\text{max}}$ with RFT + CB and linear-SVM + TB. Forbye it, we see that the *R* is high with SVM everywhere.

Figure 11 shows the improvement of the BoW + PoS approach on the conventional BoW approach. The subfigures in the top row indicate improvement w.r.t count vectorization, and in contrast, the bottom row carries information on the TF ∗ IDF vectorization. The overall observation on the improvement gives a piece of mixed information except for the TF ∗ IDF features on an average dataset, where the positive trend of improvement is steady. However, the least improvement, i.e., ≈0.8% on an average basis, is seen for the same dataset. In the same context, on average the maximum pointer of improvement (i.e., ≈3%) is found with the dataset with $\Psi_{f=\text{sum}}$.

Figure 12 shows the CM of all conventional ML algorithms, separated w.r.t feature enhancement and vectorization techniques. However, instead of multiplying the figure space three times for each of the datasets with respective final scoring functions, we have presented the aggregated-normalized CM. The colour bar on the right of Figure 13 is set to serve a specific purpose such that the maximum value is 0.5 (≈50%) which corresponds to the size of data in one class.

We maintain that the SVM + TBP with its both linear and rbf variants is the most optimal algorithm among all. This is so because linear-SVM achieved TN + TP = 0.35 + 0.4 ≈ 0.75 ≡ 75% accuracy; however, the other variant, rbf-SVM, stood second. The authors would like to maintain the performance of the RFT + CBP; 0.34 + .41 ≈ .75 also similar to the previously mentioned linear- and rbf-SVM. Forbye it, we must maintain that the competition between the SVM + CBP and RFT + CBP is near equal, but the RFT + CBP is found champion such that it has got minimum value on right diagonal (i.e., FP + FN = 0.16 + 0.09 ≈ 0.21), and in a similar context, it has got the least FN which, per se, is an additive advantage.

4.3. Analysis of DL Models and Results. The NN-based DL models are used on the dataset with the scoring function $\Psi_{f=\text{sum}}$, as it has produced the most optimal result in comparison to the remaining two scoring functions.

Table 3: Summary of gates involved in LSTM and GRU architecture.

Type of gate	Description	Utilized in
Update gate (Γ_u)	How much of the past should be remembered now?	LSTM, GRU
Relevance gate (Γ_r)	Drop previous information	LSTM, GRU
Forget gate (Γ_f)	Erase cell or not?	LSTM
Output gate (Γ_o)	How much information of cell should be revealed?	LSTM

Table 4: Architecture and variables' information the LSTM and GRU.

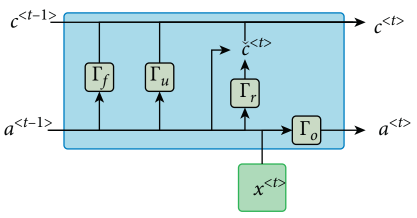
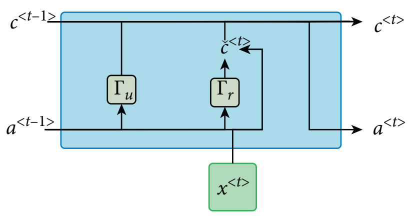
Variables	LSTM	GRU
Illustration	 $c^{<t-1>}$ $c^{<t>}$ $\check{c}^{<t>}$ Γ_f Γ_u Γ_r Γ_o $a^{<t-1>}$ $a^{<t>}$ $x^{<t>}$	 $c^{<t-1>}$ $c^{<t>}$ $\check{c}^{<t>}$ Γ_u Γ_r $a^{<t-1>}$ $a^{<t>}$ $x^{<t>}$
$\check{c}^{\langle t\rangle}$	$\tanh(W_c[\Gamma_r \odot a^{\langle t-1\rangle}, x^{\langle t\rangle}] + b_c)$	$\tanh(W_c[\Gamma_r \odot a^{\langle t-1\rangle}, x^{\langle t\rangle}] + b_c)$
$c^{\langle t\rangle}$	$\Gamma_u \odot \check{c}^{\langle t\rangle} + \Gamma_f \odot c^{\langle t-1\rangle}$	$\Gamma_u \odot \check{c}^{\langle t\rangle} + (1-\Gamma_u) \odot c^{\langle t-1\rangle}$
$a^{\langle t\rangle}$	$\Gamma_u \odot c^{\langle t\rangle}$	c^t

Table 5: Summary of DNN used in this paper.

Layer	Input unit	Output units	Activation
Input	Max # of words	32	—
Hidden	32	32	tanh
Dense/output	32	1	Sigmoid

Table 6: Configuration of DNN.

Model settings	Values
Optimizer	Adam
Loss	Binary cross-entropy
Epochs	10
Batch size	128

Table 7: A sample confusion matrix.

		Predicted	
		N	P
Actual	N	TN	FP
	P	FN	TP

Table 12 shows the metrics for the validation set only. Among the three input encoding techniques, the word2vec is found for better GDEX classification. However, the unidirectional or vanilla GRU and LSTM are found biased towards the negative class. Alternatively, in other words, the aforementioned DL networks failed to discriminate between a GDEX and bad examples and hence developed a propensity towards the negative class only. (The authors would maintain that the biasedness of unidirectional NN can be overcome by introduction of dropouts but we are afraid of doing it for the reason of being unjust to the rest of NNs employed in this work.) Moreover, this behaviour is seen for both of the dense embedding techniques word2vec and GloVe. In contrast, the bidirectional variant of these two techniques achieved approximately equal and comparatively optimal results. We maintain that word2vec with Bi-LSTM is the optimal algorithm for GDEX as it has achieved 77% accuracy (and balanced accuracy as well). Alongside it, the highest recall, i.e., 86%, is also on record for this setting. The NNs with the one-hot encodings though have shown the least but steady results.

Figure 14 shows epochwise loss and accuracy achieved in training and validation sets. We have got the typical behaviour in counting the increment in epochs; the loss, in the validation set, minimizes to an extent, and afterwards, it gets propensity to increase; in contrast, the loss continues to diminish in the training set [39, 42, 43]. We can see this behaviour in all DL models—except for the Bi-LSTM and Bi-GRU with word2vec and GloVe, which show steady performance. Furthermore, since we know that the DL is more appropriate for the largescale datasets, and currently the data employed for this experiment is comparatively smaller, we can expect a few numbers of epochs are enough for the training (or not indulging in the overfitting model on training data). In this regard, the authors maintain that the 3 epochs are enough for any of the DL-NNs used in the experiments. This is so because we see in the validation dataset that the accuracy is declining after the 3rd epoch.

Figure 13 shows the improvement in accuracy and balanced accuracy achieved by one DNN over the other networks; the quantified value of these metrics is subtracted as NN_x–NN_y provided that $x \neq y$, where x is DNN (alongside the input encoding method used in it) defined on the x-axis and y is DNN in the y-axis. The cells with the shades of red colour in the figures indicate negative improvement; in contrast, the cells with grey shades indicate improvement. The intensity of shades is directly proportional to the value of the improvement. Likewise, in the observation reported in Table 8, we found that, except for the few network comparisons, the improvement in the accuracy and balanced

Table 8: Summary of evaluation statistics used in this paper.

Name (abbreviation)	Derivation	Definition/Notes
Precision (P)	$P = (\mathrm{TP}/(\mathrm{TP}+\mathrm{FP}))$	Precision (or alternatively known as positive predictive value) reveals the ratio of TP to the documents that are predicted positive by the PS
Recall (R)	$R = (\mathrm{TP}/(\mathrm{TP}+\mathrm{FN}))$	Recall (or true positive rate) shows the right potential of the PS for predicting positive documents in the subset of all positive documents in the system
Specificity (S)	$S = (\mathrm{TN}/(\mathrm{TN}+\mathrm{FN}))$	Specificity (or true negative rate) is the exact opposite of R. It gives the potential of the PS for negative documents
F_1-measure (F)	$F = 2\cdot((P\cdot R)/(P+R))$	F_1-measure is a harmonic mean of P and R. It is important to use where the dataset is imbalanced; further, it is a strict measure, which has a propensity towards the minima of P and R [41]
Accuracy (A)	$A = ((\mathrm{TP}+\mathrm{FP})/(\mathrm{TP}+\mathrm{TF}+\mathrm{TN}+\mathrm{FN}))$	Accuracy gives the overall creditability of the PS
Balanced accuracy (BA)	$\mathrm{BA} = (R+S)/2$	Likewise F, the balanced accuracy is also a mean statistic, which gives an arithmetic mean of R and S

Table 9: Results of conventional ML algorithms. Dataset with the final scoring function $\Psi_{f=\mathrm{avg}}$.

Algo.	Vec.	Only BoW						BoW + PoS tags					
		P	R	S	F	A	BA	P	R	S	F	A	BA
k-NN	Count	55.9	55.9	45.8	55.9	57.2	50.8	62	62	53.4	62	64.8	57.7
	TF * IDF	62.2	62.2	54.6	62.2	64.6	58.4	67.7	67.7	70.5	67.7	66.1	69.1
NB	Count	63.1	63.1	56	63.1	65.6	59.5	63.5	63.5	58.4	63.5	65.4	60.9
	TF * IDF	64.3	64.3	60.4	64.3	65.9	62.4	64.9	64.9	62.2	64.9	66.1	63.6
RFT	Count	65.2	65.2	56.7	65.2	69	61	67.2	67.2	62	67.2	70	64.6
	TF * IDF	67.2	67.2	62.9	67.2	69.5	65.1	67.4	67.4	62.8	67.4	69.9	65.1
Linear-SVM	Count	67.5	67.5	62.9	67.5	70	65.2	65.5	65.5	56.4	65.5	69.7	61
	TF * IDF	68.1	68.1	64.4	68.1	70.1	66.2	69.1	69.1	66.2	69.1	70.9	67.6
Rbf-SVM	Count	66.7	66.7	60.2	66.7	69.9	63.4	66.8	66.8	60.7	66.8	69.9	63.7
	TF * IDF	68.7	68.7	65.9	68.7	70.5	67.3	69.4	69.4	66.5	69.4	71.2	67.9

Table 10: Results of conventional ML algorithms. Dataset with final scoring function $\Psi_{f=\mathrm{max}}$.

Algo.	Vec.	Only BoW						BoW + PoS tags					
		P	R	S	F	A	BA	P	R	S	F	A	BA
k-NN	Count	57.9	57.9	44.8	57.9	60.4	51.4	67.1	67.1	57.6	67.1	72.1	62.4
	TF * IDF	66.3	66.3	57.3	66.3	70.7	61.8	74	74	75.1	74	73.1	74.5
NB	Count	68.4	68.4	62.9	68.4	71.6	65.6	67.9	67.9	64.2	67.9	70	66
	TF * IDF	69.8	69.8	66.6	69.8	71.9	68.2	69.8	69.8	68	69.8	70.9	68.9
RFT	Count	70	70	62.8	70	74.8	66.4	73.4	73.4	69.5	73.4	77	71.4
	TF * IDF	75.4	75.4	72.8	75.4	78	74.1	74.3	74.3	71.3	74.3	77.2	72.8
Linear-SVM	Count	75.4	75.4	74.9	75.4	76.1	75.1	74.2	74.2	71.7	74.2	76.5	72.9
	TF * IDF	75	75	72.4	75	77.5	73.7	75.2	75.2	72.1	75.2	78.3	73.6
Rbf-SVM	Count	73.7	73.7	69.1	73.7	77.9	71.4	73.8	73.8	69.8	73.8	77.5	71.8
	TF * IDF	74.7	74.7	71.9	74.7	77.3	73.3	75.2	75.2	72.3	75.2	78.1	73.7

Table 11: Results of conventional ML algorithms. Dataset with final scoring function $\Psi_{f=\mathrm{sum}}$.

Algo.	Vec.	Only BoW						BoW + PoS tags					
		P	R	S	F_1	A	BA	P	R	S	F_1	A	BA
k-NN	Count	58.1	58.1	42.1	58.1	61.3	50.1	67.3	67.3	57.9	67.3	72.3	62.6
	TF * IDF	66.2	66.2	57.1	66.2	70.5	61.6	74.4	74.4	76.7	74.4	72.3	75.5
NB	Count	68.6	68.6	63.6	68.6	71.5	66.1	67.9	67.9	64.5	67.9	69.8	66.2
	TF * IDF	69.6	69.6	67.3	69.6	71.0	68.4	70.3	70.3	69.5	70.3	70.9	69.9
RFT	Count	70.5	70.5	63.9	70.5	75.2	67.2	73.5	73.5	69.9	73.5	76.6	71.7
	TF * IDF	74.9	74.9	73.0	74.9	76.8	73.9	74.3	74.3	71.8	74.3	76.8	73
Linear-SVM	Count	72.3	72.3	68.7	72.3	75.3	70.5	72.7	72.7	68.7	72.7	76	70.7
	TF * IDF	74.8	74.8	72.6	74.8	76.9	73.7	74.4	74.4	71.7	74.4	77	73
Rbf-SVM	Count	73.8	73.8	70.2	73.8	77.1	72.0	74.5	74.5	71.8	74.5	77.2	73.1
	TF * IDF	74.5	74.5	72.1	74.5	76.9	73.3	74.9	74.9	72.5	74.9	77.3	73.7

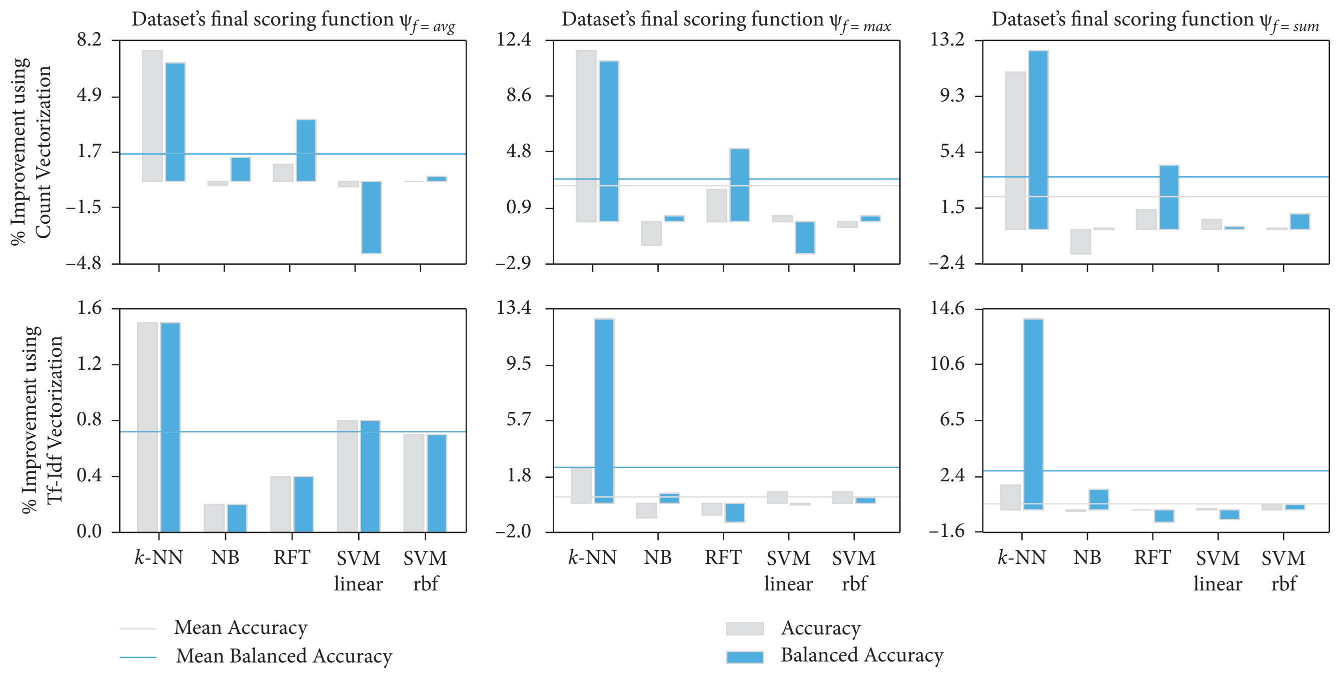


FIGURE 11: Comparative percentage improvement gained by using words + PoS tags features on only BoW features in conventional machine learning algorithms.

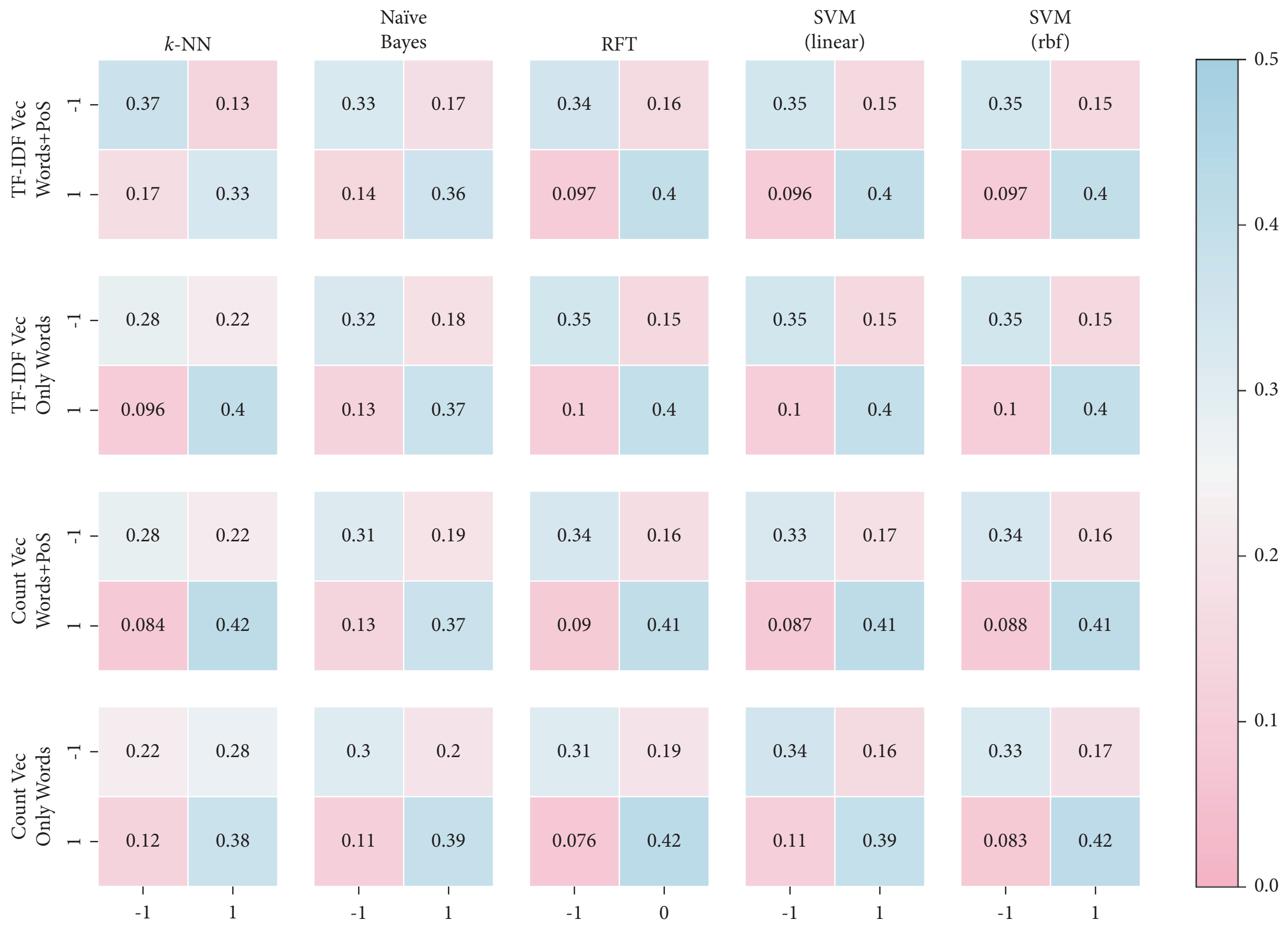


FIGURE 12: Average confusion matrices for conventional ML approaches. The confusion matrices (of datasets relating to the average, max, and sum scoring functions) are averaged, respectively, for each algorithm, vectorization, and feature set.

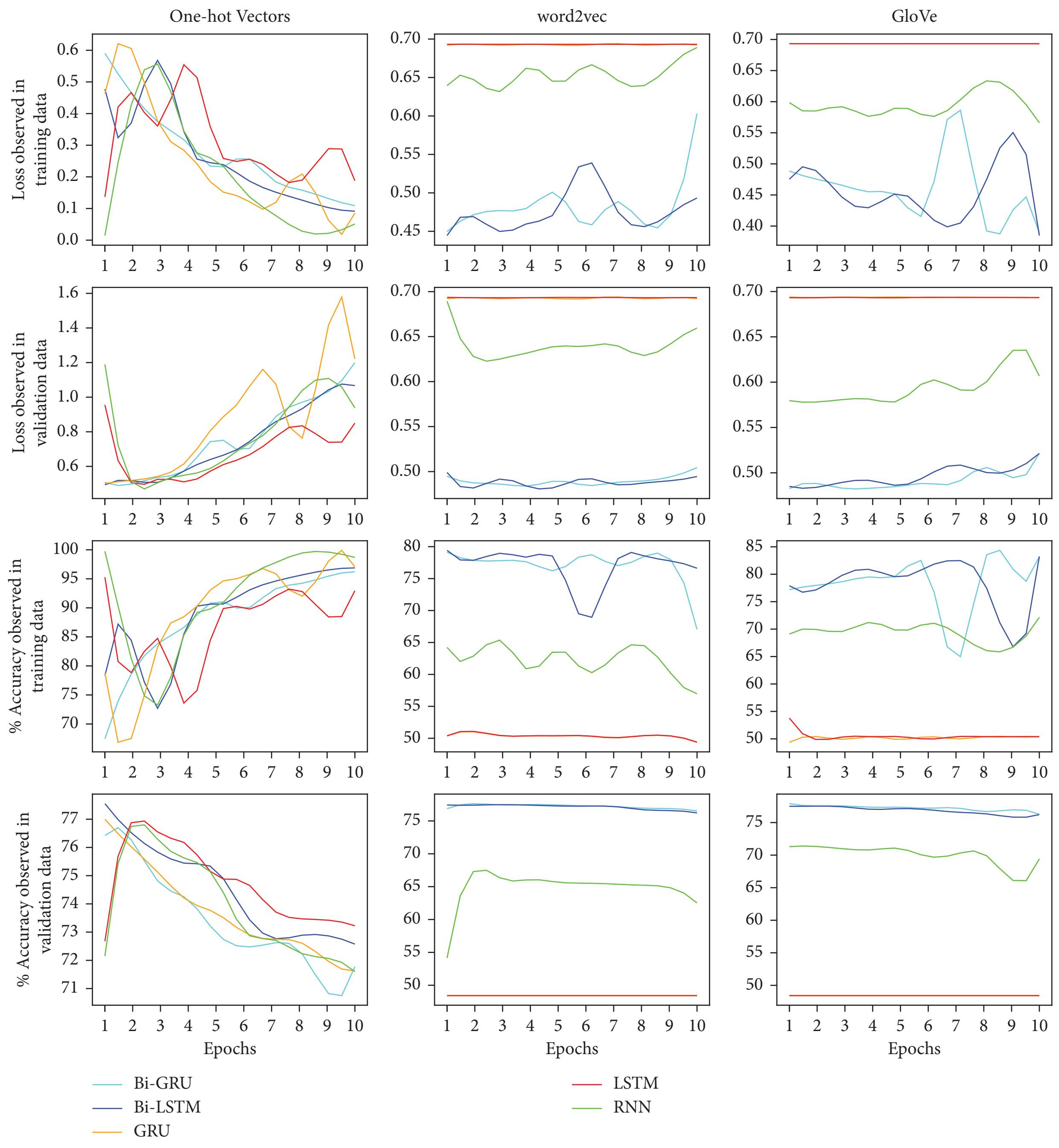


Figure 13: % improvement observed in accuracy and balanced accuracy. The values of the neural network and its encoding method (paired as *NN* + *encoding*) on the *x*-axis are subtracted from the respective pair values on the *y*-axis.

accuracy yielded in DNN are equivalently identical. We find that bidirectional DNNs with word embeddings drew a major improvement on the rest of all DNNs. In a similar context, though the highest accuracy and balanced accuracy are seen over unidirectional NN we neglect this case on the ground of biased performance shown by the unidirectional DNNs (with word embeddings). Except for the previously mentioned case, the real highest gain in accuracy and balanced accuracy is seen over RNN + word2vec; i.e., Bi-GRU and Bi-LSTM have secured ≈23% improvement with word2vec, followed by attaining ≈22% improvement by the same DNNs with GloVe. Keeping the focus on Bi-GRU and Bi-LSTM, the most optimal word embedding scheme is word2vec such that it achieved ≈4% and ≈5% improvement over vanilla one-hot encodings used for the same DNNs and ≈1% improvement over GloVe.

Observing CMs presented in Figure 15, we confirm that the bidirectional LSTM with word2vec is the most optimal NN and inputs data embedding pair for resolving the problem under study. We also maintain that, in comparison to the GloVe and word2vec, the one-hot encoding is the most underperforming input encoding scheme.

Table 12: Results of conventional DL algorithms employed for the dataset with final scoring function $\Psi_{f=\text{sum}}$.

	One-hot encoding						word2vec						GloVe					
Algo.	*P*	*R*	*S*	*F*1	*A*	BA	*P*	*R*	*S*	*F*1	*A*	BA	*P*	*R*	*S*	*F*1	*A*	BA
RNN	72.5	74.2	70	73.3	72.2	72.1	55.4	57	51	56	54	54	70.4	70.3	68.5	70.3	69.4	69.4
GRU	72.8	73	70.9	72.9	72	72	—	0	100	—	48	50	—	0	100	—	48.4	50
LSTM	74.1	72.2	73.2	73.2	72.7	72.7	—	0	100	—	48	50	—	0	100	—	48.4	50
Bi-GRU	73.8	70.2	73.5	72	71.8	71.8	77.8	77	77	78	77	77	73.1	85.5	66.5	78.8	76.3	76
Bi-LSTM	73.4	73.3	71.8	73.4	72.6	72.6	74.3	86	69	80	77	77	75	80.9	71.2	77.8	76.2	76.1

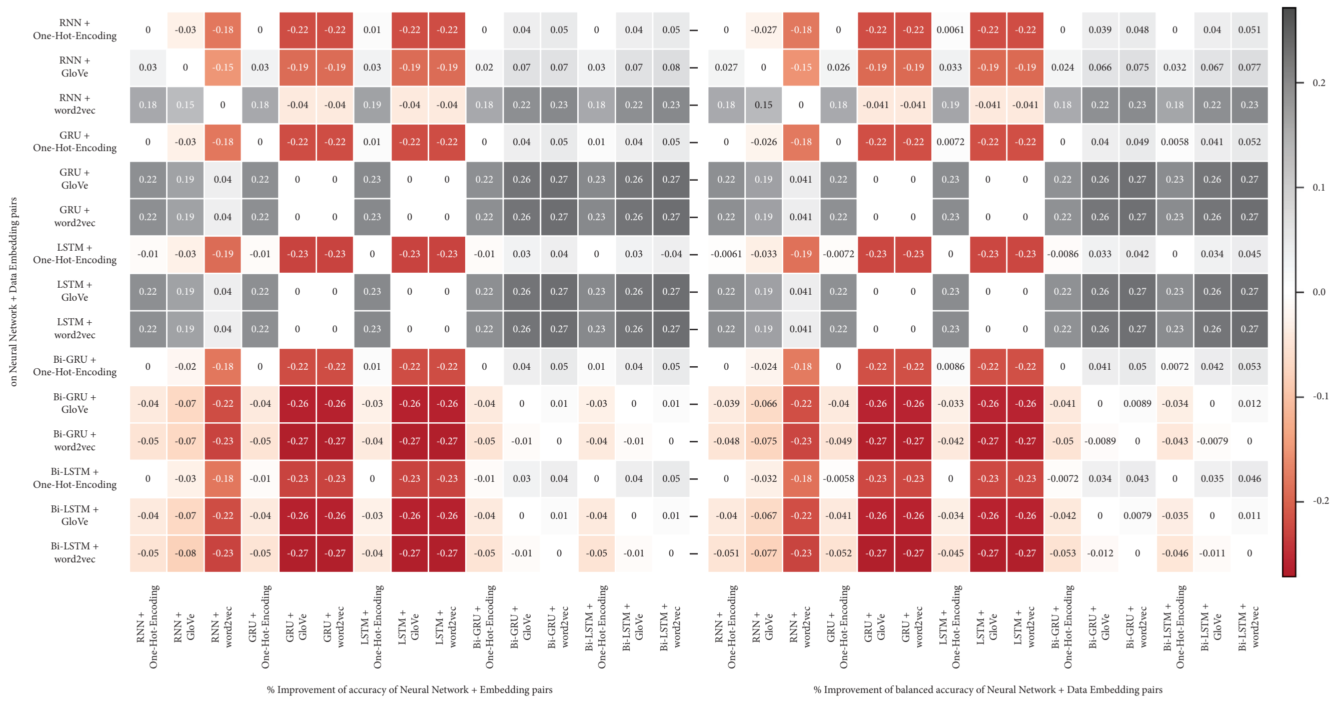


Figure 14: Epochwise insights into the loss and % accuracy w.r.t training and validation datasets.

4.4. Comparative Analysis on ML vs. DL Models. As reported in several different studies on the comparison of ML and DL models [16, 39, 43, 44], the authors of this paper reassert that the DL models outperform conventional ML models. In addition to it, we also maintain that DL-based models are revealed to attain balanced scores in accuracy and balanced accuracy. However, the DL-based unidirectional algorithms failed, which we consider specific to the problem under study; in contrast, the bidirectional DL algorithms are found the most optimal ones.

Thus, w.r.t the results compiled in Table 13, if we look at the averages of all ML models (for the dataset $\Psi_{f=\text{sum}}$) and compared them with the averaged values of DL-based models (i.e., RNN, Bi-LSTM, and Bi-GRU; leaving unidirectional LSTM and GRU due to their biasedness) then, we see only an improvement of ≈+3.56% and ≈+2.47%, respectively, in recall and balanced accuracy for GDEX classification. However, the principal reason for such small improvement lies with the lower scores of RNN in comparison to the remaining two bidirectional NNs. In contrast, the ML-based models took very little time in preprocessing and training. In a similar context, we can see the one-hot encoding turned training time longer, whereas the DL models with 300-dimensional dense word embeddings were trained in a small amount of time.

4.5. Competitive Analysis on ML and DL Models with Manual GDEX Elicitation Routines. Table 14 shows the selected examples of sentences from the test/validation set and the prediction made by the most optimal ML and DL models for them. In addition to it, we also show the GDEX rules presented in the seminal work by Kilgarriff et al. [2]. These were actually 5 rules, which are already mentioned in the literature review (see Subsection 2.2); however, rule 3 is omitted in discussion as it deals with the penalization of a sentence containing anaphors and pronouns (though there are sentences which deal the aforesaid matters, ML/DL do not explicitly deal with such penalization). Examples 1–8 show TP and TN, wherein, specifically rule#4 is false when the actual label is *bad.* Examples 9 and 10 show FP, where rule#4 is false. Examples 11 and 12 are real mistakes, as these are the FN, and the sentence not only complies with all rules but also appears to be very succinct in structure.

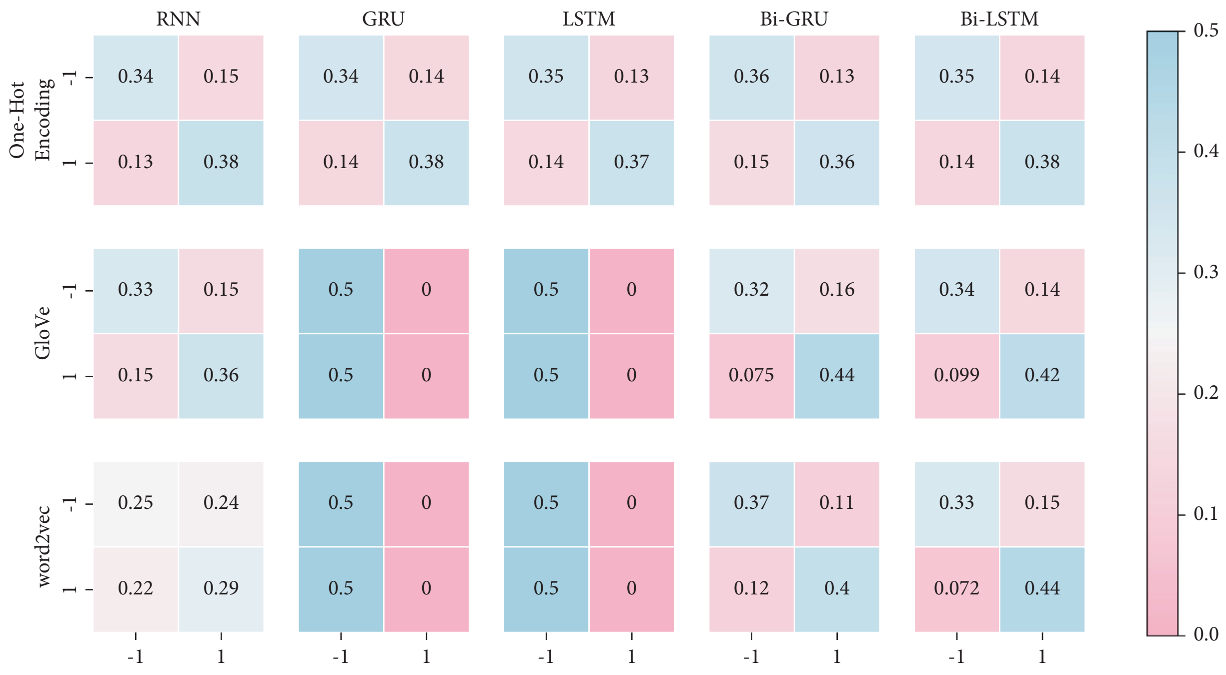


FIGURE 15: The confusion matrices (of validation set relating to the max scoring functions).

TABLE 13: Comparative averages of ML and DL performances. The $*$ in the last row indicates the number of seconds elapsed more than the average of ML-based model preprocessing and training.

Models	Evaluation metrics						Running time (in seconds)	
	P	*R*	*S*	*F*	*A*	BA	Preprocessing	Training
ML + CB	68.66	68.66	61.7	68.66	72.08	65.18	1.72	36.86
ML + TB	72	72	68.42	72	74.42	70.18	1.96	48.14
ML + CBP	71.18	71.18	66.56	71.18	74.38	68.86	9.15	33.05
ML + TBP	73.66	73.66	72.44	73.66	74.86	73.02	9.06	46.4
ML average	71.38	71.38	67.28	71.38	73.94	69.31	5.47	41.1
DL + one-hot enc.	73.23	72.57	71.77	72.9	72.7	72.17	1.23	272.33
DL + word2vec	69.17	73.33	65.67	71.33	69.33	69.33	325.68	32
DL + GloVe	72.83	78.9	68.73	75.63	73.99	73.83	32.48	31.66
DL average	71.74	74.93	68.72	73.29	72.01	71.78	119.80	112.0
Improvement in DL	0.37	3.56	1.44	1.91	-1.93	2.47	114.32 $*$	70.8 $*$

TABLE 14: Selected examples for the discussion of errors in GDEX classification.

Examples	Actual label	Prediction information		Did the example comply with rules for GDEX as defined in [2]?			
		Label	Classifier	*R*#1	*R*#2	*R*#4	*R*#5
(1) Not wanting to abandon Lori, she did nothing.	Good	Good	RFT + TBP	Yes	Yes	Yes	Yes
(2) The French have abandoned the left bank?	Bad	Bad	RFT + TBP	Yes	Yes	No	Yes
(3) She closed the cabinet door, troubled.	Good	Good	Bi-LSTM + w2v	Yes	Yes	Yes	Yes
(4) At a cabinet meeting on June 5th it is said that M.	Bad	Bad	Bi-LSTM + w2v	Yes	Yes	Yes	Yes
(5) Each dog knew its master and its call.	Good	Good	RFT + TBP	Yes	Yes	Yes	Yes
(6) The redcoats are coming, they said to each other.	Bad	Bad	RFT + TBP	Yes	Yes	No	Yes
(7) You love him very much.	Good	Good	Bi-LSTM + w2v	Yes	Yes	Yes	Yes
(8) Very. I'm so glad we have you and Jonathan	Bad	Bad	Bi-LSTM + w2v	Yes	Yes	No	No
(9) The sack and dog moved about two feet.	Bad	Good	Bi-LSTM + w2v	Yes	Yes	No	Yes
(10) For the sowing of seed see Sowing.	Bad	Good	RFT + TBP	Yes	Yes	No	Yes
(11) I see no point in telling him.	Good	Bad	Bi-LSTM + w2v	Yes	Yes	Yes	Yes
(12) He set the sack on the table.	Good	Bad	RFT + TBP	Yes	Yes	Yes	Yes

We can draw another meaningful insight into dataset curation through distant supervision. The unanimous true for rules 1 and 2 and correct assessment of rules 4 and 5 confirm the reliability of the usage of web-based data available at YD.com alongside the method for label assignment with the scoring function $\Psi_{f=\text{sum}}$ for GDEX classification and similar other problems.

5. Conclusions

This paper provides the implementation of both ML and DL models for the GDEX classification. Following the results compiled in the experiments, we conclude that the proposed methodology is accomplishable for the automation of manual GDEX elicitation routine. The dataset of 50K example is extracted with the distant supervision technique, for which the summation method is found better than vote aggregation (averaging and max) methods. For the conventional ML-based methods, the distinction of TF $*$ IDF normalization over count vectorization is revisited during experiments. Also, we have analysed that PoS features are important and better for the easy classification and discrimination of GDEX. For the DL-based models, Bi-LSTM + word2vec is the champion among the rest of all DL-based combinations.

In the future, this work could be extended by incorporating supervised learning for the GDEX elicitation against the given target word. We would also like to evaluate the current system on the attention-based DL models. At last, we would like to apply and evaluate the current technique on oriental languages such as Arabic, Persian, and Urdu—where the GDEX is considered to have historic relevance in the poetic work.

Data Availability

The models and data files can be accessed at https://github.com/MuhammadYaseenKhan/english-gdex.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study was partially financed by the two educational institutes: Mohammad Ali Jinnah University, Karachi, Pakistan, and Islamic University of Madinah, Saudi Arabia.

References

[1] B. S. Atkins and M. Rundell, *The Oxford Guide to Practical Lexicography*, Oxford University Press, Oxford, UK, 2008.

[2] A. Kilgarriff, M. Husák, K. McAdam, M. Rundell, and P. Rychlý, "Gdex: automatically finding good dictionary examples in a corpus," in *Proceedings of the XIII EURALEX International Congress*, Universitat Pompeu Fabra, Barcelona, Spain, July 2008.

[3] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, "Distant supervision for relation extraction without labeled data," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, Singapore, August 2009.

[4] W. Muhammad, M. Mushtaq, K. N. Junejo, and M. Y. Khan, "Sentiment analysis of product reviews in the absence of labelled data using supervised learning approaches," *Malaysian Journal of Computer Science*, vol. 33, no. 2, pp. 118–132, 2020.

[5] I. Pilán, E. Volodina, and R. Johansson, "Automatic selection of suitable sentences for language," in *Proceedings of the 20 Years of EUROCALL: Learning from the Past, Looking to the Future: 2013 EUROCALL Conference*, Dublin, Ireland, September 2013.

[6] I. Srdanović and I. Kosem, "GDEX for Japanese: automatic extraction of good dictionary example candidates," in *Proceedings of the GLOBALEX 2016 Lexicographic Resources for Human Language Technology Workshop Programme*, Portorož, Slovenia, May 2016.

[7] I. Kosem, K. Koppel, T. Zingano Kuhn, J. Michelfeit, and C. Tiberius, "Identification and automatic extraction of good dictionary examples: the case(s) of GDEX," *International Journal of Lexicography*, vol. 32, no. 2, pp. 119–137, 2019.

[8] A. Geyken, C. Pölitz, and T. Bartz, "Using a maximum entropy classifier to link "good" corpus examples to dictionary senses," in *Proceedings of the Electronic Lexicography in the 21st Century Conference*, Herstmonceux Castle, UK, August 2015.

[9] E. T. Jaynes, "Information theory and statistical mechanics," *Physical Review. Series II*, vol. 106, no. 4, pp. 620–630, 1958.

[10] N. Ljubešić and M. Peronja, "Electronic lexicography in the 21st century: linking lexical data in the digital age," in *Proceedings of the Electronic Lexicography in the 21st Century Conference (eLex)*, Herstmonceux Castle, UK, August 2015.

[11] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[12] R. Stanković, B. Šandrih, R. Stijović, C. Krstev, D. Vitas, and A. Marković, "SASA dictionary as the gold standard for good dictionary examples for Serbian," *Electronic lexicography in the 21st century: Smart lexicography*, 2019.

[13] K. Koppel, *Example Sentences in Estonian Learners' Dictionaries*, Institute of the Estonian Language, University of Tartu, Tartu, Estonia.

[14] S. Uprety and M. Shakya, "Role of conext clue sentences as dictionary examples," *The Journal of University Grants Commission*, vol. 6, no. 1, pp. 132–140, 2017.

[15] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, Cambridge University Press, Cambridge, UK, 2008.

[16] K. Kowsari, K. J. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: a survey," *Information*, vol. 10, no. 4, p. 150, 2019.

[17] A. R. Martinez, "Natural language processing," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 3, pp. 352–357, 2010.

[18] W. Zhang, T. Yoshida, and X. Tang, "A comparative study of TF$*$IDF, LSI and multi-words for text classification," *Expert Systems with Applications*, vol. 38, no. 3, pp. 2758–2765, 2011.

[19] E. Fix and J. L. Hodges Jr., *Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties*, USAF School of Aviation Medicine, Randolph Field, TX, USA, 1951.

[20] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[21] G. H. Jon and P. Langley, "Estimating continuous distributions in Bayesian classifiers," 2013, https://arxiv.org/abs/1302.4964a.

[22] H. Zhang, "Exploring conditions for the optimality of naïve Bayes," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 19, no. 2, pp. 183–198, 2005.

[23] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[24] C.-C. Chang and C.-J. Lin, "Libsvm," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–27, 2011.

[25] M. Y. Khan and K. N. Junejo, "Exerting 2D-space of sentiment Lexicons with machine learning techniques: a hybrid approach for sentiment analysis," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 6, pp. 599–608, 2020.

[26] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin, "Liblinear: a library for large linear classification," *Journal of Machine Learing Research*, vol. 9, no. 8, pp. 1871–1874, 2008.

[27] J. R. Firth, "A synopsis of linguistic theory, 1930–1955," *Studies in Linguistic Analysis*, Longmans, London, UK, 1957.

[28] M. Tomáš, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, GA, USA, June 2013.

[29] J. Pennington, R. Socher, and C. D. Manning, "Glove: global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, Doha, Qatar, October 2014.

[30] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[31] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[32] A. Amidi and S. Amidi, "CS 230-recurrent neural networks cheatsheet," *Stanford*, https://stanford.edu/%7Eshervine/teaching/cs-230/cheatsheet-recurrent-neural-networks, 2021.

[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[34] F. A. Gres, J. Schmidhuber, and F. Cummins, "Learning to forget: contual prediciton with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[35] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, "LSTM: a search space odessey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2015.

[36] W. Ke, D. Huang, F. Yang, and Y. Jiang, "Soft sensor development and applications based on LSTM in deep neural networks," in *Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI)*, Honolulu, HI, USA, November-October 2017.

[37] P. F. Moshiri, H. Navidan, R. Shahbazian, S. A. Ghorashi, and D. Windridge, "Using GAN to enhance the accuracy of indoor human activity recognition," 2020, https://arxiv.org/abs/2004.11228.

[38] K. Cho, B. V. Merriënboer, C. Gulcehre et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014, https://arxiv.org/abs/1406.1078.

[39] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT press, Cambridge, UK, 2016.

[40] V. Pareto, "Cours d'économie politique, Librairie Droz," 1964.

[41] S. Shaikh, M. Y. Khan, and M. S. Nizami, "Using patient descriptions of 20 most common diseases in text classification for evidence-based medicine," in *Proceedings of the Mohammad Ali Jinnah University International Conference on Computing (MAJICC'21)*, Karachi, PakistanAccepted, Karachi, Pakistan, July 2021.

[42] M. Y. Khan and T. Ahmed, "Pseudo transfer learning by exploiting monolingual corpus: an experiment on roman Urdu transliteration," in *Intelligent Technologies and Applications*, I. Bajwa, T. Sibalija, and D. Jawawi, Eds., pp. 422–431, Springer, Singapore, 2019.

[43] M. A. Nielsen, *Neural Networks and Deep Learning*, Determination Press, San Francisco, CA, USA, 2015.

[44] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, Prentice Hall, Hoboken, NJ, USA, 3rd edition, 2020.

[45] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[46] J. Schmidhuber, "Deep learning in neural networks: an overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[47] B. Farley and W. Clark, "Simulation of self-organizing systems by digital computer," *Transactions of the IRE Professional Group on Information Theory*, vol. 4, no. 4, pp. 76–84, 1954.

[48] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[49] A. G. Ivakhnenko and V. G. Lapa, *Cybernetics and Forecasting Techniques*, American Elsevier Pub. Co, Princeton, NJ, USA, 1967.

[50] P. Werbos, "Applications of advances in nonlinear sensitivity analysis," in *System Modeling and Optimization*, pp. 762–770, Springer, Berlin, Germany, 1982.

[51] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.