Article

Short Text Classification with Tolerance-Based Soft Computing Method

Vrushang Patel [1], Sheela Ramanna [2,*], Ketan Kotecha [3] and Rahee Walambe [3]

[1] Deloitte Inc., Calgary, AB T2P 0R8, Canada; vrushangpatel255@gmail.com or vrupatel@deloitte.ca
[2] Department of Applied Computer Science, University of Winnipeg, Winnipeg, MB R3B 2E9, Canada
[3] Symbiosis Institute of Technology (SIT), Symbiosis Centre for Applied Artificial Intelligence (SCAAI), Symbiosis International (Deemed University), Pune 412115, India; director@sitpune.edu.in (K.K.); rahee.walambe@sitpune.edu.in (R.W.)
* Correspondence: s.ramanna@uwinnipeg.ca

Abstract: Text classification aims to assign labels to textual units such as documents, sentences and paragraphs. Some applications of text classification include sentiment classification and news categorization. In this paper, we present a soft computing technique-based algorithm (TSC) to classify sentiment polarities of tweets as well as news categories from text. The TSC algorithm is a supervised learning method based on tolerance near sets. Near sets theory is a more recent soft computing methodology inspired by rough sets where instead of set approximation operators used by rough sets to induce tolerance classes, the tolerance classes are directly induced from the feature vectors using a tolerance level parameter and a distance function. The proposed TSC algorithm takes advantage of the recent advances in efficient feature extraction and vector generation from pre-trained bidirectional transformer encoders for creating tolerance classes. Experiments were performed on ten well-researched datasets which include both short and long text. Both pre-trained SBERT and TF-IDF vectors were used in the experimental analysis. Results from transformer-based vectors demonstrate that TSC outperforms five well-known machine learning algorithms on four datasets, and it is comparable with all other datasets based on the weighted F1, Precision and Recall scores. The highest AUC-ROC (Area under the Receiver Operating Characteristics) score was obtained in two datasets and comparable in six other datasets. The highest ROC-PRC (Area under the Precision–Recall Curve) score was obtained in one dataset and comparable in four other datasets. Additionally, significant differences were observed in most comparisons when examining the statistical difference between the weighted F1-score of TSC and other classifiers using a Wilcoxon signed-ranks test.

Keywords: sentiment classification; machine learning; tolerance near sets; transformer; news classification; Natural Language Processing

check for updates

Citation: Patel, V.; Ramanna, S.; Kotecha, K.; Walambe, R. Short Text Classification with Tolerance-Based Soft Computing Method. *Algorithms* **2022**, *15*, 267. https://doi.org/10.3390/a15080267

Academic Editors: Hui Yu, Mounim A. El Yacoubi and Mehdi Ammi

Received: 17 May 2022
Accepted: 28 July 2022
Published: 30 July 2022

1. Introduction

Text classification, also known as text categorization, is a classical problem in Natural Language Processing (NLP), which aims to assign labels to textual units such as documents, sentences, paragraphs, and queries. It has a wide range of applications including sentiment analysis, news categorization, question answering, user intent classification, spam detection, and content moderation, to name a few [1]. Popular NLP research areas where the user opinions are analyzed to detect sentiment polarity include opinion mining and sentiment analysis [2]. Polarity determination has been performed for product reviews, forums, blogs, news articles, and micro-blogs. The field of sentiment analysis is greatly aided by a rich and large source of information from platforms such as Twitter. The tasks of Twitter sentiment analysis include sentiment polarity detection, Twitter opinion retrieval, tracking sentiments over time [3], irony detection, and emotion detection [4–8]. Due to the word limit of 280 characters, micro-blogs do not contain complete sentences. Moreover, micro-blogs often contain abbreviations and noisy texts. Therefore, it needs standard pre-processing techniques such Parts-of-Speech (POS) tagging, removing of URLs, hashtags, usernames, stopwords, stemming, and spelling correction to be applied to tweets due to the nature of messages posted by users. Twitter sentiment classification identifies different polarities (e.g., positive, negative, or neutral). Classification is based on textual features which can take different forms such as (i) syntactic (e.g., n-grams, term frequencies, dependency trees), (ii) semantic (e.g., opinion and sentiment words), and usually with the aid of lexicons, (iii) stylistic (e.g., emoticons) and (iv) Twitter-specific features (e.g., hashtags and retweets). The main challenges encountered with tweets are the length of the text (max. of 280 characters) and incorrect or improper use of language. The following machine learning approaches are popular with text classification: supervised [9], semi-supervised [10] and unsupervised [11,12]. Well-known sentiment lexicons such VADER (Valence Aware Dictionary for Sentiment Reasoning) [13] were developed as an improvement over NLTK and Textblob tools. In [14–17], deep convolutional and recurrent neural networks were used in sentiment analysis.

In this paper, we present a soft computing technique-based algorithm (TSC) to classify sentiment polarities of tweets and news categories from text. The TSC algorithm is a novel supervised learning method based on tolerance near sets. Near sets theory [18,19] is a more recent soft computing methodology inspired by rough sets [20] where instead of set approximation operators used by rough sets to induce tolerance classes, the tolerance classes are directly induced from the feature vectors using a tolerance level parameter ε and a distance function. The tolerance forms of rough sets have been shown to be more effective in text categorization applications [21] where overlapping classes are induced by a *tolerance relation*. The tolerance near set-based classification algorithm was first introduced in [22]. Other applications of near sets in audio signal classification, music genre classification, and community detection in social network can be found in [23]. A theoretical treatment of the relationship between near and rough sets can be found in [24].

In this paper, we explore the effect of different vector-generation methods, tolerance class sizes, balanced and imbalanced datasets as well as a number of sentiment classes on the TSC algorithm. This paper is an extension of our previous work [25]. The extensions include experimentation on three additional text datasets, new formal definition on text-based tolerance relation, comparative work using TF-IDF vectors, additional metrics besides weighted F1, and a statistical test to observe the difference in classifiers. We have also demonstrated that with transformer-based vectors, our proposed TSC outperforms five well-known machine learning algorithms on four datasets, and it is comparable with all other datasets based on the weighted F1, Precision and Recall scores. The highest AUC-ROC score was obtained in two datasets and comparable in six other datasets. The highest ROC-PRC score was obtained in one dataset and comparable in four other datasets. Additionally, significant differences were observed in most comparisons when examining the statistical difference between the weighted F1-score of TSC and other classifiers using a Wilcoxon signed-ranks test.

The proposed sentiment/text classification pipeline is given in Figure 1 where feature vectors are generated using two pre-trained deep learning models BERT [26] and SBERT [27] shown in step 2. In step 3, a cosine distance matrix using the training set is created. This distance matrix is used to create tolerance classes in step 4. In step 5, a mean vector for each of the tolerance classes is created. This vector represents a *prototype class*. In step 6, each of these prototype classes are then labeled using the majority class of their respective tolerance class members. In step 7, for each test example (in the testing set), the cosine distance is computed from every prototype class. In step 8, the test example that has the smallest distance value to the prototype class is selected. In step 9, the label of this prototype class is then assigned to the test example. In the final step, the predicted label is then checked with original label.

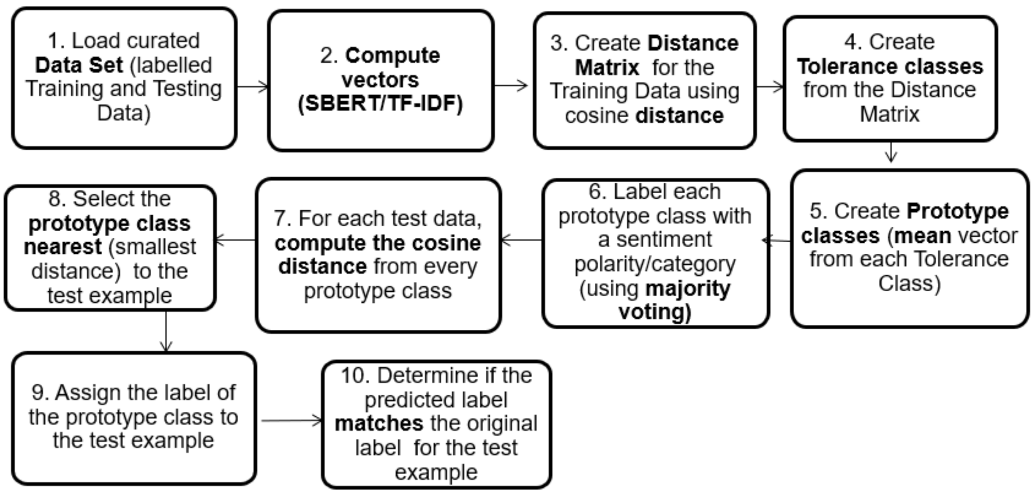

1. Load curated **Data Set** (labelled Training and Testing Data)
2. **Compute vectors (SBERT/TF-IDF)**
3. Create **Distance Matrix** for the Training Data using cosine **distance**
4. Create **Tolerance classes** from the Distance Matrix
5. Create **Prototype classes** (**mean** vector from each Tolerance Class)
6. Label each prototype class with a sentiment polarity/category (using **majority voting**)
7. For each test data, **compute the cosine distance** from every prototype class
8. Select the **prototype class nearest** (smallest distance) to the test example
9. Assign the label of the prototype class to the test example
10. Determine if the predicted label **matches** the original label for the test example

Figure 1. High-level process flow of the proposed TSC algorithm.

This paper is organized as follows. In Section 2, we introduce formal definitions for text-based tolerance relation and an illustration of sample tolerance classes. In Section 3, we present the datasets used in this work as well as the proposed supervised TCS algorithm. In Section 4, we discuss our findings in terms of the weighted F1-score measure using both TF-IDF and transformer-based vectors on all the ten datasets followed by the concluding remarks in Section 5.

2. Tolerance Relation: Definition

Near sets are essentially disjoint sets that contain objects with similar descriptions provided the intersection of the sets is nonempty. The basic structure which underlies near set theory is a perceptual system which consists of perceptual objects [19]). Near sets are characterized by a perceptual system, a nearness relation and a near set [24]. In this work, we define a nearness relation using Definition 1:

Definition 1. Text-based Tolerance Relation $\cong_{\mathcal{T},\epsilon}$
Let $\langle T, F\rangle$ *be a universe of nonempty set of objects* T *and* F *be the feature set. Let* $\mathcal{T} \subseteq F$ *where* $\mathcal{T}$ *represents textual features. A tolerance space* $\langle T, \cong_{\mathcal{T},\epsilon}\rangle$ *isdefinedas*:

$$\cong_{\mathcal{T},\epsilon} = \{(t_i, t_j) \in T \times T : dist(t_i, t_j) \leq \varepsilon\} \tag{1}$$

where *dist* is the cosine distance given in Equation (2). The tolerance relation $\cong_{\mathcal{T},\epsilon}$ induces a tolerance class TC where ε is a user-defined tolerance level.

$$dist(t_i, t_j) = 1 - \frac{\phi(t_i).\phi(t_j)}{\|\phi(t_i)\|\|\phi(t_j)\|} \tag{2}$$

In other words, given a set of text (objects) T, where $t_i \in T, i \in N$, each tweet or text t_i can be represented as a k-dimensional word vector $\phi(t_i)$ where text similarity is measured using the cosine distance measure.

Remark 1. *Our universe of text* T *described by the set of vectors* ϕ *is spread amongst tolerance classes with a tolerance level* ε *for semantic textual similarity. An illustration with examples can be found in [23].*

3. Materials and Methods

We have created a subset of ten selected benchmark datasets which are a mix of long and short words (indicated by words per sentence) with a varying number and sizes of sentiment classes (positive, negative, neutral and irrelevant). Due to memory limitations, some large datasets were trimmed and only a subset was used in our experiments. *COVID-Sentiment* is a manually labeled dataset which is a subset derived from [28] using Tweets ID for 1 April 2020 and 1 May 2020. We extracted 47,386 tweets with the help of Twitter API. The tweets in languages other than English (ex: French, Hindi, Mandarin, and Portuguese) were removed. Extensive pre-processing of 29,981 English language tweets from the original dataset such as removal of HTML tags, @Username, Hashtags, URLs, and incorrect spellings were also performed. A total of 8003 hand-labeled tweets were prepared for experimentation. The Python regex module and NLTK stemming and lemmatization were used in pre-processing before generating vector embedding for this dataset. *The U.S. Airline Sentiment* dataset consisted of 14,621 tweets and the pre-processed dataset of 13,000 tweets were used after the removal of duplicate and short tweets. For the *IMDB Movie Review*, we used a subset of 22,000 reviews of the original dataset consisting of 50,000 reviews and for the *SST-2 dataset*, the original dataset included 69,723 phrases and only 16,500 were used. For the *Sentiment140* dataset, a subset of 16,000 tweets from 1,600,000 were used. The *SemEval 2017* includes 62,671 tweets in the original dataset. We were able to use only 20,547 tweets in our experiments due to memory limitation. The *AG-News* dataset contains 496,835 categorized news articles from more than 2000 news sources. Only the four largest classes from this corpus were selected to construct this dataset. The title and description fields were included in this dataset. These two columns were used as features for classification. To generate vector embeddings, these columns are combined into a single column. It contains four categories of news: "World", "Sports", "Business" and "Science". We used 3000 samples from each category as our training set for our experiments. This dataset did not require any further pre-processing because it did not contain any grammar or spelling mistakes.

3.1. Materials

Table 1 gives details of the dataset. For each dataset, the total size of training and testing sets is given in column 3. In addition, columns 4, 5, 6, and 7 give the size of each sentiment class used for training and testing (except for the AG-news dataset). The last column shows the words per sentence (WPS) for each dataset. Only one dataset *Sanders corpus* has four sentiment classes with imbalanced distribution. Datasets *UCI Sentence, Sentiment140, SST-2, IMDB Movie Review* have two sentiment classes with a fairly balanced distribution. Two datasets *COVID-Sentiment, U.S. Airline Sentiment, SemEval 2017* have three sentiment classes with imbalanced distribution. The *20 Newsgroups* dataset is a common benchmark used for evaluating the performance of text classification algorithms. The dataset, introduced in [29], contains approximately 20,000 newsgroup posts and this dataset is partitioned (nearly) evenly across 20 different newsgroups organized into broader categories: computers, recreation, religion, science, sale, and politics. Scikit-learn was used to prepare the training and testing datasets to remove noisy data. This dataset has 10,314 training samples and 782 testing samples with 209 words per document. Table 2 gives the number of documents used in the training process and the size of the tolerance classes for each newsgroup category with the best value for $\varepsilon = 0.19$.

Table 1. Dataset Information.

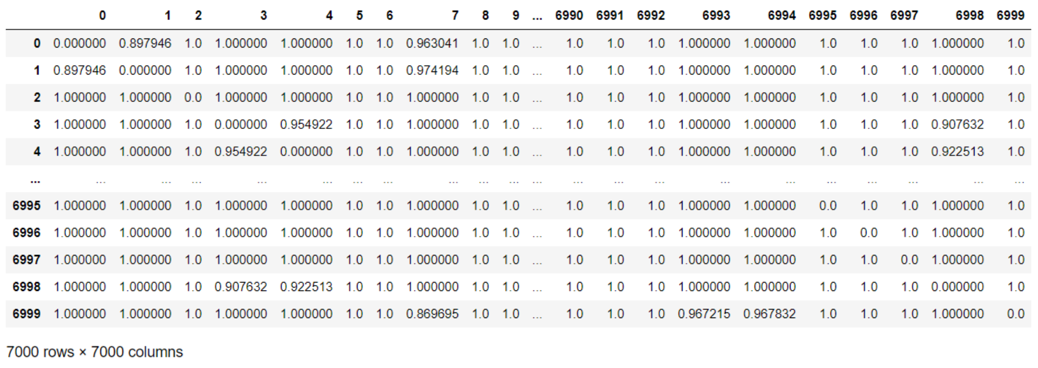
Dataset	Type	Size	Positive	Negative	Neutral	Irrelevant
COVID-Sentiment	Train	7000	22.02%	30.35%	47.63%	-
	Test	1003	23.53%	37.29%	39.18%	-
U.S. Airline Sentiment	Train	12,000	16.79%	61.02%	22.19%	-
	Test	1000	13%	67.5%	19.5%	-
IMDB Movie Review	Train	20,000	50.27%	49.73%	-	-
	Test	2000	50.35%	49.65%	-	-
SST-2	Train	15000	55.37%	44.63%	-	-
	Test	1500	55.53%	44.47%	-	-
Sentiment140	Train	15000	50%	50%	-	-
	Test	1000	50%	50%	-	-
SemEval 2017	Train	17001	40.67%	15%	44.33%	-
	Test	3546	41.54%	15.76%	42.70%	-
Sanders corpus	Train	4059	10.24%	11.38%	45.26%	33.12%
	Test	1015	9.85%	10.54%	47.68%	31.93%
UCI Sentence	Train	2700	49.11%	50.89	-	-
	Test	300	58%	42%	-	-

Table 2. 20-Newsgroups dataset for best ε value of 0.19.

News Category	#Train Documents	# Tolerance Classes
alt.atheism	442	230
comp.graphics	534	238
comp.os.ms-windows.misc	528	498
comp.sys.ibm.pc.hardware	540	445
comp.sys.mac.hardware	527	428
comp.windows.x	561	400
misc.forsale	535	312
rec.autos	526	342
rec.motorcycles	548	189
rec.sport.baseball	554	319
rec.sport.hockey	551	332
sci.crypt	537	437
sci.electronics	547	146
sci.med	551	285
sci.space	525	272
soc.religion.christian	542	494
talk.politics.guns	499	380
talk.politics.mideast	511	335
talk.politics.misc	421	248
talk.religion.misc	335	130

3.2. Methods

In this section, we present our proposed Tolerance Sentiment Classifier (TSC) in terms of the two Algorithms 1 and 2. The TSC algorithm was implemented using Python on a 16 GB RAM, Nvidia RTX 2060 GPU, 512 GB SSD machine using SBERT base vectors (1×768 dimensional vectors). We considered mean and median values for determining the prototype class vectors for the TSC algorithm. In addition, we experimented using TF-IDF vectors with the TSC algorithm for all the datasets. However, the classification results with TF-IDF vectors were unsatisfactory primarily because the cosine distance values started to converge to one and hence resulted in a number of the same values in the distance matrix shown in Figure 2, which restricts the use of TF-IDF vectors.



	0	1	2	3	4	5	6	7	8	9	...	6990	6991	6992	6993	6994	6995	6996	6997	6998	6999
0	0.000000	0.897946	1.0	1.000000	1.000000	1.0	1.0	0.963041	1.0	1.0	...	1.0	1.0	1.0	1.000000	1.000000	1.0	1.0	1.0	1.000000	1.0
1	0.897946	0.000000	1.0	1.000000	1.000000	1.0	1.0	0.974194	1.0	1.0	...	1.0	1.0	1.0	1.000000	1.000000	1.0	1.0	1.0	1.000000	1.0
2	1.000000	1.000000	0.0	1.000000	1.000000	1.0	1.0	1.000000	1.0	1.0	...	1.0	1.0	1.0	1.000000	1.000000	1.0	1.0	1.0	1.000000	1.0
3	1.000000	1.000000	1.0	0.000000	0.954922	1.0	1.0	1.000000	1.0	1.0	...	1.0	1.0	1.0	1.000000	1.000000	1.0	1.0	1.0	0.907632	1.0
4	1.000000	1.000000	1.0	0.954922	0.000000	1.0	1.0	1.000000	1.0	1.0	...	1.0	1.0	1.0	1.000000	1.000000	1.0	1.0	1.0	0.922513	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6995	1.000000	1.000000	1.0	1.000000	1.000000	1.0	1.0	1.000000	1.0	1.0	...	1.0	1.0	1.0	1.000000	1.000000	0.0	1.0	1.0	1.000000	1.0
6996	1.000000	1.000000	1.0	1.000000	1.000000	1.0	1.0	1.000000	1.0	1.0	...	1.0	1.0	1.0	1.000000	1.000000	1.0	0.0	1.0	1.000000	1.0
6997	1.000000	1.000000	1.0	1.000000	1.000000	1.0	1.0	1.000000	1.0	1.0	...	1.0	1.0	1.0	1.000000	1.000000	1.0	1.0	0.0	1.000000	1.0
6998	1.000000	1.000000	1.0	0.907632	0.922513	1.0	1.0	1.000000	1.0	1.0	...	1.0	1.0	1.0	1.000000	1.000000	1.0	1.0	1.0	0.000000	1.0
6999	1.000000	1.000000	1.0	1.000000	1.000000	1.0	1.0	0.869695	1.0	1.0	...	1.0	1.0	1.0	0.967215	0.967832	1.0	1.0	1.0	1.000000	0.0

7000 rows × 7000 columns

Figure 2. Distance Matrix for TF-IDF Vectors.

Vector Embeddings with SBERT: Sentence-BERT (SBERT) is a modification of the pre-trained BERT network that uses siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using cosine-similarity. SBERT is fine-tuned on SNLI [30] and the Multi-Genre NLI [31] data, which creates sentence embeddings and significantly outperforms other state-of-the-art sentence embedding methods such as InferSent [32] and and Universal Sentence Encoder [33] in terms of accuracy.

Training Phase: Representative Class Generation Algorithm 1: In this phase, given a tolerance level ε, tolerance classes are induced from the training set vectors using the cosine distance, and the representative of each tolerance class is computed as the mean value of the feature vector. The polarity (or category) of the representative vector is determined based on majority voting.

Testing Phase: Polarity Assignment Algorithm 2: In the classification phase, TSC uses the representative class vectors generated in the training phase and their associated polarity/text category. The $computeCosineDist$ function calculates the cosine distance between each test set vector and all the representative class vectors. The $DeterminePolarity$ function chooses the representative class that is *closest* to the test set vector and assigns the polarity of the representative to the test set vector. In the training phase, the complexity of $computeCosineDist$ function is $\mathcal{O}(n^2)$. The complexity of *generatetolerantpairs* function is $\mathcal{O}(n)$. In the testing phase, the complexity of $DeterminePolarity$ function is $\mathcal{O}(n)$.

In the *testing phase*, the cosine distance is computed for each vector in the testing set by comparing with all representative vectors obtained in the training phase. The test set vector with the lowest cosine distance value is assigned the polarity (or category) of the representative.

Algorithm 1: Training Phase: Generating class representative vectors

```
Input  : $TV = \{TV_1, \ldots, TV_M\}$ , // Transformer Vectors for training
         $\varepsilon > 0$, // Tolerance level parameter
Output: $(NT, \{(R_1, TextCat_1), \ldots, (R_{NT}, TextCat_{NT})\})$
         NT is the size of the Tolerance class set
// Create a distance matrix for all training vectors based on the function defined in
  Equation 2
for $p \leftarrow 1$ to $M$ do
    for $q \leftarrow p + 1$ to $M$ do
        computeCosineDist($TV_p, TV_q, Cos_{pq}$)
//a. Create object pairs satisfying the tolerance relation defined in Equation 1.
b. Create a neighborhood of training vectors.
c. Generate tolerance class set T where $TC_i$ is one tolerance class member induced by the
  tolerance relation
d. Determine the majority polarity element in each $T_i$
e. Compute representative class vectors
for $i \leftarrow 1$ to $M$ do
    for $j \leftarrow i + 1$ to $M$ do
        ObjectPairs $\leftarrow$ generatetolerantpairs($\mathrm{Cos}_{ij}, \varepsilon$);
        $N_i \leftarrow$ createobjectneighbour(ObjectPairs, $i, TV$); // Compute the
          neighbourhood $N_i$ of $i^{th}$ training vector $TV_i$
        for all, $o_1, o_2 \in N_i$ do
            if $o_1, o_2 \in ObjectPairs$ then
                $TC_i \leftarrow \{o_2\}$; // Include $o_2$ from $N_i$ into $TC_i$
    $T \leftarrow T \cup \{TC_i\}$; // T is set of all tolerance classes
    $TextCat_i \leftarrow$ computeMajorityPol($T_i$); // For each $T_i$, determine polarity/category
      by majority voting
$NT \leftarrow |T|$; // Number of tolerance classes in T // End of the process of generating
  set T which is set of all tolerance classes.
//For each tolerance class in T, generate a representative class vector and its
  polarity/category.
$\{(R_1, TextCat_1), \ldots, (R_{NT}, TextCat_{NT})\} \leftarrow$ GenerateClassRepresentative($NT$);
```

Algorithm 2: Testing Phase: Assigning Sentiment Polarities

```
Input  :$\varepsilon > 0$, // Tolerance level parameter
         , $NT$ // Size of the tolerance class set $T$
         , $TV' = \{TV'_1, \ldots, TV'_M\}$, // transformer vectors for testing
         $\{(R_1, TextCat_1), \ldots, (R_{NT}, TextCat_{NT})\}$ // Representative class vectors
generated in the training phase and their associated polarities
Output: $(TV' = \{(TV'_1, TextCat_1), \ldots, (TV'_M, TextCat_M)\})$ // Transformer
         vectors with assigned polarities/categories
for $i \leftarrow 1$ to $M$ do
    for $j \leftarrow i + 1$ to $NT$ do
        computeCosineDist($TV'_i, R_j, \mathrm{Cos}_{ij}$);
$TV' \leftarrow$ DeterminePolarity($Cos_{ij}$) // Computes min. distance and assigns polarity
  to the test set vector
```

4. Results and Discussion

In this section, we discuss the performance of the TSC algorithm. Figure 3 gives the weighted F1-score for all datasets for various tolerance values using the mean value (TSC-mean) for the prototype class vector. The range of tolerance values is from 0.08 to 0.38. The TSC algorithm performs best with the UCI sentence dataset and has the worst performance with the COVID-Sentiment dataset. With the 20-Newsgroups dataset, our proposed algorithm shows a steep improvement with the tolerance value between 0.12 and 0.19. Figure 4 gives the median value (TSC-median) for the prototype class vectors. Note that even though the relative performance is similar with all datasets, the most noteworthy difference is with the *U.S. Airline and IMDB* datasets. Since the overall results with the mean value are slightly better in terms of the weighted F1-score for all the datasets, we used the mean value for the TSC algorithm (TSC-mean) in all our subsequent experiments.

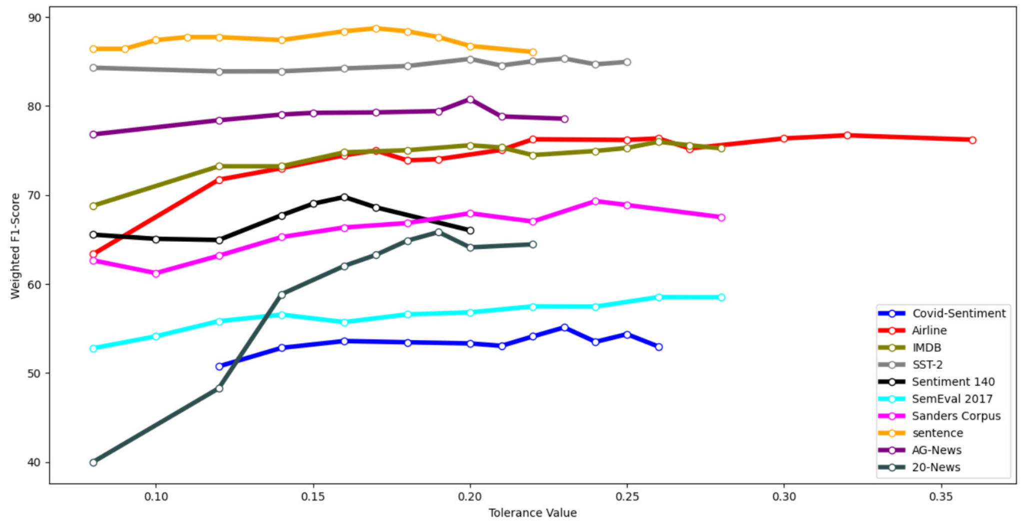


Figure 3. TSC Algorithm: Weighted F1-score for all datasets using mean-value (TSC-mean) prototype vectors for different ε values.

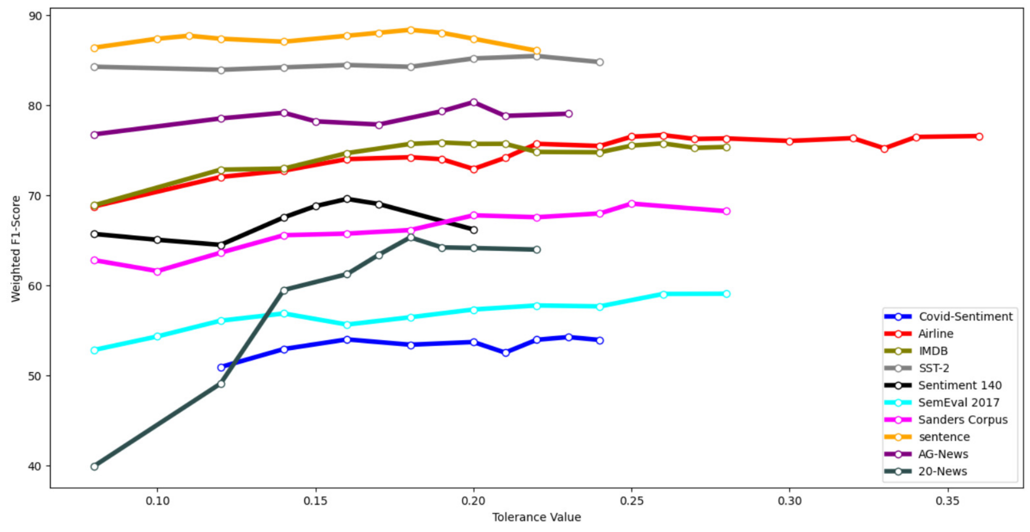


Figure 4. TSC Algorithm: Weighted F1-score for all datasets using median-value (TSC-median) prototype vectors for different ε values.

Table 3 shows the number of *tolerance* classes for the best tolerance value (column 2) for each dataset. The TSC algorithm generates these classes as described in Algorithm 1. It should be noted that the SST-2 and UCI sentence datasets have an approximately similar number of tolerance classes and have two sentiment classes. It should be noted that ε values range from 0.16 to 0.32. Other datasets with three and four sentiment classes do not generate balanced tolerance classes.

Random Forest (RF) [34], Maximum Entropy (ME), Support Vector Machine (SVM) [35], Stochastic Gradient Decent (SGD) [36] and Light Gradient Boosting Machine (LGBM) [37] classifier implementations in Scikit-learn (https://scikit-learn.org/stable/) (accessed on 1 May 2022) with the following parameters were used. For the RF classifier, 100 trees and gini index were used to determine the quality of split. The minimum and maximum samples were set to 2 and 1, respectively, to split an internal node of the tree, and the maximum number of features was set to 27 (square root of the size of the vector). Bootstrap samples were used when building trees and the random_state parameter was set to 42. For the ME (logistic regression) classifier, the l2 penalty term with the stopping criteria set to 10^{-4} was used. The RBF kernel was used for the SVM classifier with a kernel cache size = 200 MB, gamma set to scale, C value set to 1 and l2 penalty term. The hinge loss function was used in the SGD classifier with default values for other parameters (penalty l2, alpha set to 0.0001, maximum iterations set to 1000 and learning rate set to optimal). Since the loss is hinge, this classifier is a linear SVM. For the LGBM classifier, the max_leaf_nodes was set to 31, and the learning_rate, n_estimators, min_child_weight, and min_child_samples were set to 0.1, 100, 10^{-3}, and 20, respectively.

Table 4 shows the AG-News dataset with a well-balanced tolerance class distribution for the four categories (TC-World, TC-Sports, TC-Business and TC-Science) for $\varepsilon = 0.2$. It should be noted all algorithms perform (third best) on this dataset. The 20-Newsgroups dataset has the highest number of categories among all other datasets. It has twenty sentiment classes and a fairly balanced tolerance class distribution as shown in Table 2. Due to better semantic similarity of its vectors, the performance of TSC on this dataset is better than the COVID-sentiment and SemEval 2017 datasets.

Table 3. Best tolerance ε value with sizes of tolerance classes for the positive, negative, neutral and irrelevant text polarities.

Dataset	**ε Value**	**TC-Positive**	**TC-Negative**	**TC-Neutral**	**TC-Irrelevant**
COVID-Sentiment	0.23	249	1136	2590	-
U.S. Airline	0.32	2009	8202	1234	-
IMDB	0.26	6979	12,956	-	-
SST-2	0.23	6501	5531	-	-
Sentiment140	0.16	926	1969	-	-
SemEval 2017	0.26	6583	1755	5797	-
Sanders corpus	0.24	205	390	1415	1518
UCI Sentence	0.17	576	557	-	-

Table 4. Best ε value for the TSC algorithm and tolerance class size for AG-News dataset.

Dataset	**ε Value**	**TC-World**	**TC-Sports**	**TC-Business**	**TC-Science**
AG-News	0.2	1900	2607	2563	2398

Table 5 gives the weighted F1-score for all datasets with the TF-IDF-based ML algorithms. The size of TF-IDF vectors depends on the vocabulary of that dataset, which means having longer sentences results in better vocabulary for the TF-IDF approach, which is an advantage over transformer vectors. While building vocabulary with TF-IDF, the frequency of the words was considered to compute the vectors. We considered default parameters of TF-IDF to build the vocabulary. The minimum and maximum document frequency was set to the default value of 1.0. The results show that TF-IDF-based ML algorithms give the best results in longer sentences or document-level classification tasks such as IMDB and 20-newsgroups datasets. Table 6 gives the experimental results with the transformer-based vectors. Here, our proposed TSC algorithm performs best in the U.S. Airline, IMDB, UCI SST-2 and 20-Newsgroup datasets and is comparable with the COVID-Sentiment, SST-2 and Sentiment 140 datasets. It can be observed that balanced tolerance classes are an indication of good semantic similarity between vectors generated by the transformer model. This can be seen with SST-2 and UCI sentence datasets that have approximately a similar number of tolerance classes with a weighted F1-score of over 85%. Another observation is that these two datasets contain only two sentiment classes. On the other hand, the TSC algorithm performs very well with the 20-Newsgroups datasets.

Table 5. TF-IDF-based weighted F1-score (rounded) results for five classifiers. Best results are in bold face.

Dataset	RF	ME	SVM	SGD	LGBM
COVID-Sentiment	**67**	59	62	62	58
U.S. Airline	78	79	79	80	**80**
IMDB	83	87	**89**	89	86
SST-2	82	82	82	82	74
Sentiment140	71	**74**	72	72	73
SemEval 2017	64	64	65	63	**65**
Sanders corpus	74	74	75	**75**	69
UCI Sentence	70	77	76	76	63
AG-News	78	84	84	**85**	82
20-Newsgroups	62	73	74	**76**	66

Table 6. SBERT vector-based weighted F1-score (rounded) results for six classifiers. Best results are in bold face.

Dataset	TSC-Mean	RF	ME	SVM	SGD	LGBM
COVID-Sentiment	55	44	57	57	**57**	56
U.S. Airline	**77**	77	77	77	75	77
IMDB	**76**	69	73	73	72	72
SST-2	85	85	85	**86**	85	85
Sentiment140	70	68	**72**	72	66	70
SemEval 2017	60	54	**64**	63	63	60
Sanders corpus	69	70	**76**	74	76	75
UCI Sentence	**89**	84	86	87	87	83
AG-News	82	79	88	81	**88**	83
20-Newsgroups	**66**	41	58	52	52	53

Tables 7 and 8 give the weighted precision and recall scores for all the tested classifiers. Based on this score, the proposed TSC algorithm performs best in the UCI, SST-2, IMDB and 20-Newsgroups datasets. These results mirror the values obtained with the weighted F1-score except with the U.S. Airline dataset.

Table 9 gives the AUC-ROC score for all the tested classifiers. A pair-wise comparison of all combinations (ovo) with the weighted average score parameter was used for obtaining the results. Based on the AUC-ROC score, the proposed TSC algorithm performs best in the UCI and SST-2 datasets that have an approximately similar number of tolerance classes and two classes. It is also noteworthy that the UCI and AGnews datasets have the best separability (90%), and IMDB (88%), 20-Newsgroups (87%) and SST-2 (85%) have scores above 84% based on the tested classifiers. Since SGD is a linear SVM, for most datasets, the weighted F1, Precision and Recall scores are either similar or identical.

Table 7. SBERT vector-based Weighted Precision score for six classifiers. Best results are in bold face.

Dataset	TSC-Mean	RF	ME	SVM	SGD	LGBM
COVID-Sentiment	0.58	**0.60**	0.59	0.59	0.59	**0.60**
U.S. Airline	0.77	0.75	**0.79**	0.76	0.78	0.77
IMDB	**0.76**	0.69	0.73	0.73	0.72	0.72
SST-2	**0.85**	0.83	**0.85**	**0.85**	**0.85**	**0.85**
Sentiment140	0.71	0.69	**0.72**	**0.72**	0.71	0.71
SemEval 2017	0.59	0.58	**0.65**	0.64	0.63	0.62
Sanders corpus	0.70	**0.78**	0.76	0.74	0.77	**0.78**
UCI Sentence	**0.90**	0.87	0.86	0.87	0.87	0.85
AG-News	0.81	0.80	**0.85**	0.82	0.82	0.83
20-Newsgroups	**0.67**	0.42	0.59	0.53	0.60	0.51

Table 8. SBERT vector-based Weighted Recall score for six classifiers. Best results are in bold face.

Dataset	TSC-Mean	RF	ME	SVM	SGD	LGBM
COVID-Sentiment	0.56	0.50	**0.57**	**0.57**	**0.57**	**0.57**
U.S. Airline	0.78	0.76	**0.80**	0.77	0.78	0.79
IMDB	**0.76**	0.69	0.73	0.73	0.72	0.72
SST-2	**0.85**	0.83	**0.85**	**0.85**	**0.85**	**0.85**
Sentiment140	0.70	0.68	**0.72**	**0.72**	0.67	0.70
SemEval 2017	0.59	0.57	**0.64**	0.63	0.63	0.61
Sanders corpus	0.70	0.76	0.77	0.75	0.76	**0.78**
UCI Sentence	**0.89**	0.84	0.86	0.87	0.87	0.83
AG-News	0.81	0.80	**0.85**	0.82	0.82	0.83
20-Newsgroups	**0.66**	0.44	0.58	0.52	0.55	0.50

Table 9. SBERT vector-based AUC-ROC score for six classifiers. Best results are in bold face.

Dataset	TSC-Mean	RF	ME	SVM	SGD	LGBM
COVID-Sentiment	0.65	**0.71**	0.68	0.70	0.70	0.67
U.S. Airline	0.75	0.77	0.76	0.77	0.77	**0.79**
IMDB	0.76	0.83	0.83	0.86	**0.88**	0.87
SST-2	**0.85**	0.82	0.81	0.82	0.82	0.73
Sentiment140	0.70	0.71	**0.74**	0.72	0.73	0.73
SemEval 2017	0.66	0.71	0.71	0.71	0.70	**0.72**
Sanders corpus	0.77	0.77	0.78	0.79	**0.80**	0.76
UCI Sentence	**0.90**	0.72	0.78	0.77	0.76	0.67
AG-News	0.87	0.86	**0.90**	**0.90**	**0.90**	0.88
20-Newsgroups	0.82	0.81	0.86	0.86	**0.87**	0.82

Table 10, gives the AUC-PRC score for all the tested classifiers. The loss function for the SGD classifier was changed from *hinge* (default) to *modified_huber* to enable probabilistic outputs (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html accessed on 1 May 2022). Since all other metrics for SVD were based on the hinge loss function, the results for this classifier (indicated in blue) could be omitted in the overall analysis. Based on the AUC-PRC score, the proposed TSC algorithm performs best with the IMDB dataset and is comparable with the UCI, SST-2, Sentiment140 and COVID-Sentiment datasets.

In terms of the three reported metrics (weighted F1, AUC-ROC, AUC-PRC) from Tables 6, 9 and 10 respectively, overall, the proposed TSC algorithm performs best with mostly balanced datasets having two sentiment classes (binary classification) with the U.S. Airline dataset being the exception. TSC performs poorly with two highly imbalanced datasets having more than sentiment classes (i.e., Sanders corpus and SemEval 2017). The weighted F1 score is computed only on predicted classes, whereas the AUC scores reflect the performance of a classifier over a range of values (prediction score). In comparison with other classifiers, the proposed TSC algorithm does better in more datasets with the AUC-ROC score. However, if we examine the overall AUC scores ≥ 80, TSC gives better performance with five datasets using the AUC-PRC score as compared to four datasets with the AUC-ROC score. Another point to note is that the size of tolerance classes for negative sentiment is larger the size of tolerance classes for the positive sentiment in two of these datasets (i.e., IMDB and US Airline). This result leads us to conclude that perhaps balanced tolerance classes may not be a significant factor and may depend on the vector generation method. Table 11 gives the results of the Wilcoxon signed-ranks test based on the weighted F1-score of the classifiers using a two-sided test with $\alpha = 0.05$ to test the null hypothesis that there is no difference between the TSC-mean and other classifiers. The results are a pair-wise test on all datasets. Based on these results, the null hypothesis can be rejected (i.e., there is a difference between between the classifiers based on weighted F1-score).

Table 10. SBERT vector-based AUC-PRC score for six classifiers. Best results are in bold face.

Dataset	**TSC-Mean**	**RF**	**ME**	**SVM**	SGD	**LGBM**
COVID-Sentiment	0.58	0.54	0.61	**0.64**	0.43	0.60
U.S. Airline	0.80	0.83	**0.87**	**0.87**	0.71	0.86
IMDB	**0.84**	0.78	0.81	0.82	0.76	0.81
SST-2	0.90	0.90	0.93	**0.94**	0.85	0.92
Sentiment140	0.75	0.74	0.79	**0.79**	0.53	0.77
SemEval 2017	0.62	0.60	0.69	**0.70**	0.59	0.66
Sanders corpus	0.72	0.84	0.84	0.85	0.58	**0.86**
UCI Sentence	0.95	0.90	0.95	**0.96**	0.83	0.92
AG-News	0.85	0.86	0.90	**0.91**	0.75	0.90
20-Newsgroups	0.42	0.38	0.62	**0.64**	0.35	0.54

Table 11. Wilcoxon signed-rank test.

Classifiers	**Test-Statistic**	***p*-Value**
TSC-mean—RF	1	0.017
TSC-mean—ME	15	0.6736
TSC-mean—SVM	21	0.8582
TSC-mean—SGD	21	0.8583
TSC-Mean—LGBM	7.5	0.5270

5. Conclusions

In this paper, we present a tolerance near sets-based (TSC) algorithm to classify sentiment polarities as well as news categories from text. We have introduced formal definitions and illustrate our method using tolerance classes for generating representative vectors. The performance of the TSC algorithm was tested with ten well-researched text classification datasets including a manually labeled dataset from Twitter about opinions on COVID-19 across the globe. The curated datasets include long and short words, a mix of sentiment classes as well as balanced and imbalanced datasets. We have analyzed the effect of TF-IDF and the transformer vector-generation method (SBERT) on the performance of the TSC algorithm and five classical ML algorithms. We have demonstrated that the TSC algorithm performs well in seven out of ten datasets using weighted F1, Precision and Recall scores. The highest AUC-ROC score was obtained in two datasets and comparable in six other datasets. The highest AUC-PRC score was obtained in one dataset and comparable in four other datasets. Additionally, significant differences were observed in most comparisons when examining the statistical difference between the weighted F1-score of TSC and other classifiers using a Wilcoxon signed-rank test. As future work, optimization of the TSC algorithm in terms of computing tolerance classes is a natural extension of the current work by using parallel computing with CUDA to compute the distance matrix. This is necessary in order to extend this work to the original dataset, which is larger than the curated datasets, and to test whether balanced tolerance classes are indeed a definitive factor in classification performance. Another factor to consider is to use some filtering criteria to control the size of the vocabulary when computing TF-IDF vectors specific to the datasets. This could be important when comparing with TSC where a fixed size vector was used. Experiments with other word-embedding methods (e.g., Word2Vec, GloVe, fastText) as well as examining the impact of the bag-of-words representation dimensionality will be explored.

Author Contributions: Conceptualization, Validation, Writing—original draft draft preparation, Supervision, S.R.; Software, Data Curation, Validation, Investigation, V.P.; Writing—review and editing, K.K. and R.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the NSERC Discovery Grant # 194376.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All datasets used were publicly available. The COVID-sentiment dataset can be made available upon request.

Acknowledgments: This research was supported by the NSERC Discovery Grant # 194376. The authors are deeply grateful to Cenker Sengoz for his help in implementing the AUC-PR code for the TSC algorithm. The authors wish to thank the University of Winnipeg's Faculty of Graduate Studies for the UW President's Distinguished Graduate Studies Scholarship and the anonymous reviewers for their constructive suggestions.

Conflicts of Interest: The manuscript is approved for publication by all authors. The work described is original research that has not been published previously and is not under consideration for publication elsewhere. This is part of Vrushang Patel's MSc Thesis on Short Text Classification with Tolerance Near Sets, defended at the University of Winnipeg in 2021.

References

1. Young, T.; Hazarika, D.; Poria, S.; Cambria, E. Recent trends in deep learning based natural language processing. *IEEE Comput. Intell. Mag.* **2018**, *13*, 55–75. [CrossRef]
2. Pang, B.; Lee, L. *Opinion Mining and Sentiment Analysis. Foundations and Trends (R) in Information Retrieval*; Now Publishers: Boston, MA, USA, 2008; Volume 2, pp. 1–135. [CrossRef]
3. Bollen, J.; Pepe, A. Modeling public mood and emotion: Twitter sentiment and socioeconomic phenomena. In Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM'11), Barcelona, Spain, 17–21 July 2011; pp. 450–453.
4. Giachanou, A.; Crestani, F. Like It or Not: A Survey of Twitter Sentiment Analysis Methods. *ACM Comput. Surv.* **2016**, *49*, 46. [CrossRef]
5. Kouloumpis, E.; Wilson, T.; Moore, J. Twitter sentiment analysis: The good the bad and the omg! In Proceedings of the International AAAI Conference on Web and Social Media, Barcelona, Spain, 17–21 July 2011; Volume 5, pp. 538–541.
6. Saif, H.; He, Y.; Alani, H. Alleviating data sparsity for twitter sentiment analysis. In Proceedings of the 2nd Workshop on Making Sense of Microposts: Big Things Come in Small Packages at the 21st International Conference on theWorld Wide Web (WWW'12), CEUR Workshop Proceedings (CEUR-WS.org), Lyon, France, 16 April 2012; pp. 2–9.
7. Saif, H.; Fernandez, M.; He, Y.; Alani, H. On stopwords, filtering and data sparsity for sentiment analysis of twitter. In Proceedings of the LREC 2014, Ninth International Conference on Language Resources and Evaluation, Reykjavik, Iceland, 26–31 May 2014; pp. 810–817.
8. Terrana, D.; Augello, A.; Pilato, G. Automatic unsupervised polarity detection on a twitter data stream. In Proceedings of the 2014 IEEE International Conference on Semantic Computing, Newport Beach, CA, USA, 16–18 June 2014; pp. 128–134.
9. Pang, B.; Lee, L.; Vaithyanathan, S. Thumbs up? Sentiment classification using machine learning techniques. *arXiv* **2002**, arXiv:cs/0205070.
10. Turney, P.D. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. *arXiv* **2002**, arXiv:cs/0212032.
11. Paltoglou, G.; Thelwall, M. Twitter, MySpace, Digg: Unsupervised Sentiment Analysis in Social Media. *ACM Trans. Intell. Syst. Technol.* **2012**, *3*, 1–19. [CrossRef]
12. Jiménez-Zafra, S.M.; Martín-Valdivia, M.T.; Martínez-Cámara, E.; Ureña-López, L.A. Combining resources to improve unsupervised sentiment analysis at aspect-level. *J. Inf. Sci.* **2016**, *42*, 213–229. [CrossRef]

13. Hutto, C.; Gilbert, E. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In Proceedings of the International AAAI Conference on Web and Social Media, Ann Arbor, MI, USA, 1–4 June 2014; Volume 8, pp. 216–225.
14. Dong, L.; Wei, F.; Tan, C.; Tang, D.; Zhou, M.; Xu, K. Adaptive Recursive Neural Network for Target-dependent Twitter Sentiment Classification. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Baltimore, MD, USA, 22–27 June 2014; pp. 49–54.
15. Tang, D.; Wei, F.; Yang, N.; Zhou, M.; Liu, T.; Qin, B. Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014—Proceedings of the Conference, Baltimore, MD, USA, 23–25 June 2014; Volume 1, pp. 1555–1565.
16. Yenter, A.; Verma, A. Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysis. In Proceedings of the 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), New York, NY, USA, 19–21 October 2017; pp. 540–546.
17. Jianqiang, Z.; Xiaolin, G. Deep Convolution Neural Networks for Twitter Sentiment Analysis. *IEEE Access* **2018**, *6*, 23253–23260. [CrossRef]
18. Peters, J. Near sets. General theory about nearness of objects. *Appl. Math. Sci.* **2007**, *1*, 2609–2629.
19. Peters, J. Near sets. Special theory about nearness of objects. *Fundam. Inform.* **2007**, *75*, 407–433.
20. Pawlak, Z. *Rough Sets: Theoretical Aspects of Reasoning About Data*; Springer Science and Business Media: Berlin/Heidelberg, Germany, 1992
21. Ho, T.B.; Nguyen, N.B. Nonhierarchical document clustering based on a tolerance rough set model. *Int. J. Intell. Syst.* **2002**, *17*, 199–212. [CrossRef]
22. Poli, G.; Llapa, E.; Cecatto, J.; Saito, J.; Peters, J.; Ramanna, S.; Nicoletti, M. Solar flare detection system based on tolerance near sets in a GPU-CUDA framework. *Knowl.-Based Syst. J.* **2014**, *70*, 345–360. [CrossRef]
23. Patel, V. Short Text Classification with Tolerance Near Sets. Master's Thesis, University of Winnipeg, Winnipeg, MB, Canada, 2021. [CrossRef]
24. Wolski, M. Perception and classification. A Note on Near sets and Rough sets. *Fundam. Inform.* **2010**, *101*, 143–155. [CrossRef]
25. Patel, V.; Ramanna, S. Tolerance-based short text Sentiment Classifier. In *International Joint Conference on Rough Sets, Lecture Notes in Artificial Intelligence*; Springer: Berlin, Germany, 2021; pp. 239–245.
26. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
27. Reimers, N.; Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Hong Kong, China, 3–7 November 2019.
28. Chen, E.; Lerman, K.; Ferrara, E. Tracking social media discourse about the covid-19 pandemic: Development of a public coronavirus twitter data set. *JMIR Public Health Surveill.* **2020**, *6*, e19273. [CrossRef] [PubMed]
29. Lang, K. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*; Elsevier: Amsterdam, The Netherlands, 1995; pp. 331–339.
30. Bowman, S.R.; Angeli, G.; Potts, C.; Manning, C.D. A large annotated corpus for learning natural language inference. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; Association for Computational Linguistics: Lisbon, Portugal, 2015; pp. 632–642. [CrossRef]
31. Williams, A.; Nangia, N.; Bowman, S. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*; Association for Computational Linguistics: New Orleans, LA, USA, 2018; pp. 1112–1122. [CrossRef]
32. Conneau, A.; Kiela, D.; Schwenk, H.; Barrault, L.; Bordes, A. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 7–11 September 2017; Association for Computational Linguistics: Copenhagen, Denmark, 2017; pp. 670–680. [CrossRef]
33. Cer, D.; Yang, Y.; Kong, S.Y.; Hua, N.; Limtiaco, N.; John, R.S.; Constant, N.; Guajardo-Céspedes, M.; Yuan, S.; Tar, C.; et al. Universal sentence encoder. *arXiv* **2018**, arXiv:1803.11175.
34. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
35. Chang, C.; Lin, C.J. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.* **2011**, *2*, 1–27. [CrossRef]
36. Bottou, L.; Bousquet, O. *The Tradeoffs of Large Scale Learning*; MIT Press: Cambridge, MA, USA, 2007; Volume 20, pp. 1–9.
37. Tianqi, C.; Carlos, G. XGBoost. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016. [CrossRef]