WEC: Weighted Ensemble of Text Classifiers

Ashish Upadhyay, Tien Thanh Nguyen, Stewart Massie, John McCall
School of Computing Science and Digital Media
Robert Gordon University
Aberdeen, UK
{a.upadhyay, t.nguyen11, s.massie, j.mccall}@rgu.ac.uk

***Abstract*—Text classification is one of the most important tasks in the field of Natural Language Processing. There are many approaches that focus on two main aspects: generating an effective representation; and selecting and refining algorithms to build the classification model. Traditional machine learning methods represent documents in vector space using features such as term frequencies, which have limitations in handling the order and semantics of words. Meanwhile, although achieving many successes, deep learning classifiers require substantial resources in terms of labelled data and computational complexity. In this work, a weighted ensemble of classifiers (WEC) is introduced to address the text classification problem. Instead of using majority vote as the combining method, we propose to associate each classifier's prediction with a different weight when combining classifiers. The optimal weights are obtained by minimising a loss function on the training data with the Particle Swarm Optimisation algorithm. We conducted experiments on 5 popular datasets and report classification performance of algorithms with classification accuracy and macro F1 score. WEC was run with several different combinations of traditional machine learning and deep learning classifiers to show its flexibility and robustness. Experimental results confirm the advantage of WEC, especially on smaller datasets.**

***Index Terms*—Text Classification, Ensemble Method, Ensemble of Classifiers, Multiple Classifiers, Particle Swarm Optimisation.**

I. Introduction

Text classification is one of the most popular tasks of Natural Language Processing (NLP) which involves assigning a sentence/document one category from a list of predefined categories. There are various real-world applications ranging from classifying a review's sentiment to classifying news/research articles into various topics in online libraries [1]. In recent years, there has been much research on this topic which mainly focuses on two aspects: representation of the text; and choice of classifier to approximate the relations between text representation and categories.

Before the wave of Deep Neural Networks (DNNs), statistical representations of text, such as n-gram and Bag-of-Words based Term Frequency (TF) and Term Frequency-Inverse Document Frequency (TF-IDF) features were frequently used. In order to utilise these representations, machine learning algorithms, such as Naïve Bayes (NB) and Support Vector Machine (SVM) are frequently employed as the classification model. Although these traditional classification models can achieve good performance, they are only able to use the presence of a word in the document and do not address the order or semantics of the words which is a drawback of traditional machine learning approaches.

Word Embedding algorithms such as word2vec and GloVe [2], [3] introduced a new approach to NLP in which the vectored representation of words can be used to capture the semantics. The dense representation of a word in high dimensional space tries to group the words with similar meaning in the same cluster and increases the distance from the words with dissimilar meaning. These word embeddings are fed into different types of DNNs such as Convolutional Neural Networks (CNNs) [4], [5], Long short-term memory (LSTM) [6] and Recurrent Neural Networks (RNNs) [7] with various backbone architectures to learn the relationship between the representation and its associated class label in the training data. Several advanced versions of Word Embeddings such as ELMo [8] and BERT [9] capture the context of a word in the sentence by using language models (LM) to generate deep contextualised representations of full sentences using pre-trained LMs with RNN or Transformer architectures.

Despite many successes with DNNs compared to the traditional algorithms on text classification problems, DNNs have their own drawbacks in terms of high resource requirements in relation to the amount of labelled data and computational training time. In many real-world applications where acquiring labelled data is an integral part of a process, employing deep learning algorithms in the initial phase is not effective due to the lack of labelled data. In this work, we propose a novel weighted ensemble of different algorithms and statistical text representations to classify text data. We construct an ensemble of text classifiers in which each classifier is obtained by training a different learning algorithm (traditional or deep learning algorithms) on a specific representation (i.e. set of features) extracted from the text sentence or document. The selected classifiers are combined to obtain the final collaborated prediction. In the proposed combining method, each classifier puts different weights on its predictions which reflect its contribution to the collaborated prediction. We propose to search for the optimal combining weights by minimising a 0-1 loss function on the training data.

The main contributions of our work are:

- introduction of Weighted Ensemble for Text Classification (WEC), a novel ensemble model based on a weighted combining method to address the text classification problem;

- introduction of a new search method for the optimal combining of weights by minimising the 0-1 loss function on the training data; and
- demonstrating effectiveness by experimentally showing that WEC is better than some well-known benchmark algorithms on a number of datasets [1].

The rest of the paper is organised as follows. In section 2 we discuss some related work on existing text classification approaches, focusing on deep learning and ensemble methods. In section 3 we introduce our proposed method. Section 4 has details of our experiment set up while results and discussion follow in section 5. In section 6 we give our conclusions and propose some modifications for future work.

II. Related Work

A. Approaches for text classification

In recent years, there has been significant development in deep learning techniques applied to text classification. The state-of-the-art for learning the approximation between text representation and categories has mostly shifted from using traditional machine learning to deep learning methods.

Authors in [4] presented a CNN architecture which uses the word2vec word embedding to represent text data. In this work, a multi-channel convolution layer was applied on top of the word embedding generated from a pre-trained word2vec model and then a maxpool layer is applied to all the convolution filters. Finally, the convolution layers are concatenated and flattened into a vector for applying the softmax function on the final layer. Another version of CNN was presented in the work [5], where a sentence is encoded on the character level instead of word-level to represent the textual data. In this work, authors identified 70 different characters (including alphabets, digits and special characters) for encoding and then represented a sentence with one-hot encoded vector with 70 elements.

With the introduction of the new idea of contextualised word vectors representation [8], [10], the pre-trained LMs are used to represent the words in a sentence using their context. These methods generally take the character level encoded words and apply a bidirectional-LSTM LM to learn the contextual representation of the words. Following this idea, Howard and Ruder [11], presented a technique of transfer learning for text classification where a LM is trained unsurprisingly on a huge corpus of text followed by the domain-specific fine-tuning of the LM. Finally, this fine-tune LM is frozen with a softmax layer on top followed by fine-tuning by unfreezing the LM layer by layer from penultimate to the first layer on domain-specific data. The idea behind this is to have different learning rates for different layers. This method was introduced by the name of Universal Language Model Fine Tuning or ULMFiT which outperformed many state-of-the-art methods on more than five standard datasets.

Vaswani et.al. [12] presented a new deep architecture called Transformers which only uses a combination of attention systems and feed-forward neural networks without any convolutional or recurrent element to train the deep learning models on huge datasets in comparatively less training time. Using the Transformer architecture, the pre-trained LMs such as Open AI's GPT [13] and Google's BERT [9] are used for generating a contextualized vector representation of words in a sentence. These pre-trained LM can be used for any downstream task with just a few modifications, such as adding the softmax at the top layer when applied to text classification. These works have largely given improved performance on various NLP tasks [9], [13], including text classification on which they have established themselves as current state-of-the-arts on many datasets ranging from topic classification to sentiment analysis.

Although these methods perform well on a lot of tasks and datasets, they come with a disadvantage of needing huge labelled datasets for the training phase. When sufficient labelled data is not available during the training phase, these methods tend to perform poorly compared to other traditional machine learning algorithms.

B. Ensemble methods for text classification

Ensemble learning is a learning mechanism by combining a set of learners i.e. classifiers to obtain better result than using each individual learner. In general, ensemble learning can be categorised into two main types namely homogeneous ensemble and heterogeneous ensemble [14], [15]. In homogeneous ensemble, many new training sets are generated from the original training data. One learning algorithm then trains the set of learners on these new training sets. Several well-known homogeneous ensemble methods are Bagging, Boosting, Random Subspace, and Random Forest. Meanwhile, in heterogeneous ensemble, different learning algorithms train learners on the original training data and final prediction is made by combining the output of these learners. This type of ensemble focuses on designing the combining algorithm that combines the outputs of the learners. One well-known heterogeneous ensemble approach is Stacking which trains the combining algorithm on the predictive outputs of training observations [16].

Several examples of ensemble methods in text classification using different representations or learning algorithms are as follows. In the work [17], the authors proposed the use a RNN layer on top of the CNN + max pooling layers supporting the idea of capturing both local and global textual semantics from a sentence. In [18], linguistic features such as parts of speech and word senses with TF-IDF vectors, were combined with several learning methods such as Naive Bayes and SVM. Authors in [19] analysed the use of bigram, unigram, TF and TF-IDF features with different ensemble learning algorithms (e.g., Boosting, Random Subspace, and Bagging) combined with other classification algorithms such as Naive Bayes (NB), Support Vector Machine (SVM), and k-nearest neighbours (k-NN). In [20], an ensemble classification method is presented which utilises the static classifier selection with majority voting error and the multi-objective differential evolution algo-

[1]The intial codebase can be found at: https://github.com/panditu2015/WEC

rithm for sentiment analysis. The drawback of these techniques is that they only combine either different learning algorithms on same representation or different representations with same learning algorithms for their ensemble. On the other hand, our heterogeneous stacking ensemble method utilises different representations combined with different learning algorithms at the same time.

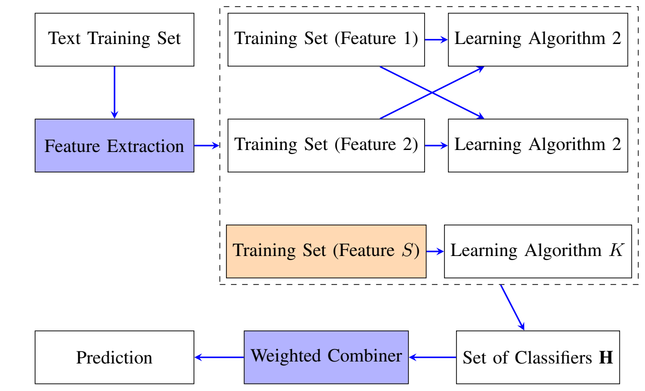
III. Proposed Method

A. General Descriptions

Let $\mathbf{D}$ be the training data, i.e. the text documents with class labels. Each class label belongs to the set of labels $Y = \{y_1, y_2, \cdots, y_M\}$. Let $\mathbf{H} = \{h_i\}$ be the ensemble of classifiers in which each classifier is a function from the feature space to Y which returns a M-dimension vector $S_i(\cdot) = [s_{i,1}(\cdot), s_{i,2}(\cdot), \cdots, s_{i,M}(\cdot)]$ where $s_{i,m}(\cdot)$ is the support that classifier h_i gives to the hypothesis that a sample belongs to the class label y_m. The $s_{i,m}(\cdot)$ is considered under the assumption $s_{i,m}(.) \in [0,1]$ and $\sum_{m=1}^{M} s_{i,m}(\cdot) = 1$ for each i [21]. In ensemble methods for the classification problem, one combining algorithm C is applied to $S_i(.)$ to obtain the combined hypothesis $\tilde{\mathbf{h}}(\cdot) = C\{S_i(\cdot)\}$. The hypothesis $\tilde{\mathbf{h}}(\cdot)$ will be used to predict a class label for samples.

The proposed method is presented in Fig 1. It is recognised that there are some well-known feature representations proposed for the text classification problem such as TF, TF-IDF, and Word Embedding. From the given text training set, we apply several feature extraction methods to get the new training sets associated with each of these feature representations. The K learning algorithms will train on these new training sets to obtain the classifiers. It is noted that some representations are only suitable for particular learning algorithms. For example, TF and TF-IDF are a 1-D feature representation that is suitable for traditional machine learning algorithms. Meanwhile, the Word Embedding is a 2-D feature representation, which is suitable for DNNs. Assume that we can train the L classifiers from the set of S types of feature representations and K learning algorithms. After obtaining the classifiers, we combine their outputs for the final collaborated prediction. Normally, the outputs of the classifiers are combined in which the role of each classifier is treated equally in the combination [16]. In fact, different learning algorithms use different approaches to train the classifiers, resulting in the differences in their classifiers' outputs. Some classifiers which perform well on specific tasks should have higher contributions than those that perform poorly. In this study, we propose a weighted combining method on the outputs of the classifiers for the text classification. Lets denote the weights as $\mathbf{w} = w_{im}$ in which $w_{im} \in [0,1]$ is the weight of classifier h_i putting on the combining result on class label y_m. By using the different weights among the different classifiers, we can set the contributions of the classifiers to the final combination, thus intuitively expecting higher prediction performance.

The question that arises from the proposed method is how to search for a suitable combining weight $\mathbf{w}$ for each situation. We formulate the optimisation problem which we can solve to



There are two modules namely Feature Extraction and Weighted Combiners are noticed. The learning algorithm 1 and 2 can train the classifiers on the training set associated with Feature 1 and 2. Meanwhile, the learning algorithm K can only work on the training set associated with the feature S (in orange color).

Fig. 1. The proposed weighted ensemble of classifiers for text classification.

find the optimal value for $\mathbf{w}$. Starting from the support matrix $\mathbf{S}$ in (1) including the vector of supports of all classifiers to all the training observations [22] ($\mathbf{S}$ is a matrix of N-row as number of training observations and $M \times L$-columns as the concatenation of the vector of supports of each classifier for each observation), we formulate the combined hypothesis from $\mathbf{S}$ and $\mathbf{w}$ by (2):

$$\mathbf{S} = \begin{bmatrix} s_{1,1}(\mathbf{x}_1) \cdots s_{1,M}(\mathbf{x}_1) \cdots s_{L,1}(\mathbf{x}_1) \cdots s_{L,M}(\mathbf{x}_1) \\ \ddots \\ s_{1,1}(\mathbf{x}_N) \cdots s_{1,M}(\mathbf{x}_N) \cdots s_{L,1}(\mathbf{x}_N) \cdots s_{L,M}(\mathbf{x}_N) \end{bmatrix} \tag{1}$$

$$\tilde{\mathbf{h}}(\mathbf{x}_n) : \mathbf{x}_n \in y \text{ if } y = \text{argmax}_{y_m, m=1 \cdots M} \frac{1}{L} \sum_{i=1}^{L} w_{im} s_{i,m}(\mathbf{x}_n) \tag{2}$$

Since $\mathbf{x}_n$ is a training observation, its ground truth is available. Based on the comparison between the prediction given by hypothesis $\tilde{\mathbf{h}}(\mathbf{x}_n)$ and the ground truth $\hat{y}$, we can compute the loss-value on $\mathbf{x}_n$:

$$\mathcal{L}_{0-1}(\mathbf{w}, \mathbf{x}_n) = 1 - \left\| \tilde{\mathbf{h}}(\mathbf{x}_n) = \hat{y_n} \right\| \tag{3}$$

in which $\|\cdot\|$ returns 1 if the condition is true, otherwise returns 0. This loss function is for the classification error rate which is one of the most popular performance metrics in the literature [21]–[23]. The loss on the training set associated with $\mathbf{w}$ is given by:

$$\mathcal{L}_{0-1}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}_{0-1}(\mathbf{w}, \mathbf{x}_n) \tag{4}$$

The optimal combining weight is obtained by minimising the loss value in (4) for $w_{im} \in [0,1], i = 1, \cdots, L; m = 1, \cdots, M$. The optimisation problem is given by:

$$\begin{gathered} \text{min}_{\mathbf{w}=\{w_{im}\}} \mathcal{L}_{0-1}(\mathbf{w}) \\ \text{s.t. } w_{im} \in [0,1], i = 1, \cdots, L; m = 1, \cdots, M \end{gathered} \tag{5}$$

B. Optimisation

In this study, we use the Particle Swarm Optimisation (PSO) algorithm to search for the optimal weights used to combine classifiers. PSO is stochastic population-based algorithm, originally introduced by Kennedy and Eberhart [24], [25], inspired by the emergent motion of a flock of birds searching for food. In comparison to other optimisation algorithms, PSO has some advantages. As a member of the family of evolutionary computation, it is well suited to handle non-linear, non-convex spaces with non-differentiable, discontinuous objective functions. PSO can work with diverse types of variables including continuous, discrete and integer types. In comparison to other evolutionary computation-based optimisation methods, PSO requires a fewer number of function evaluations, while leading to better or the same quality of results [26]. PSO, in addition, can be efficiently parallelized to reduce computational cost.

The main concept of PSO is a particle involving two components: a position vector which refers to a potential solution (candidate) for the optimisation problem and a velocity vector. To search for the optimal solution, five steps on the particles are conducted:

Step 1: We first initialise a set of positions $\mathbf{w}_i^{(0)}(i = 1, \cdots, nPoP)$ where $nPoP$ is the number of candidates in each generation and velocities $\mathbf{v}_i^{(0)}$ randomly distributed throughout the design space bounded by specified limits. Normally, the $\mathbf{w}_i^{(0)}$ and $\mathbf{v}_i^{(0)}$ are generated from the uniform distribution as $\mathbf{w}_i^{(0)} \sim U[b_l, b_u]$, $\mathbf{v}_i^{(0)} \sim U[-|b_u - b_l|, |b_u - b_l|]$. We also initialise the best know position $\mathbf{p}_i^{(0)}$ by $\mathbf{w}_i^{(0)}$ and update the swarm's best-known position $\mathbf{p}_g^{(0)}$ based on the comparison between the value of the objective functions $f(\mathbf{p}_i^{(0)})$ and $f(\mathbf{p}_g^{(0)})$.

Step 2: At the t^{th} iteration $(1 \leq t \leq maxT)$, we update the position of each particle using its previous position and its updated velocity vector as in (6) and (7):

$$\mathbf{w}_i^{(t)} = \mathbf{w}_i^{(t-1)} + \mathbf{v}_i^{(t)} \tag{6}$$

$$\mathbf{v}_i^{(t)} = a\mathbf{v}_i^{(t)} + C_1 r_1(\mathbf{p}_g^{(t-1)} - \mathbf{w}_i^{(t-1)}) + C_2 r_2(\mathbf{p}_g^{(t-1)} - \mathbf{w}_i^{(t-1)}) \tag{7}$$

in which a is inertia weight used to set up the balance between the abilities of global and local search in PSO, C_1 and C_2 are the social and cognitive attraction indicating how much confidence it has in the swarm or in itself respectively, and r_1 and r_2 are two random number drawn from a uniform distribution over $[0, 1]$.

Step 3: We evaluate the objective function values $f(\mathbf{w}_i^{(t)})$. In this study, the objective function is the $0-1$ loss defined in (4).

Step 4: The optimum particle position $\mathbf{p}_i^{(t)}$ at the current iteration and the global optimum particle position $\mathbf{p}_g^{(t)}$ are updated based on the comparison between $f(\mathbf{w}_i^{(t)})$ and $f(\mathbf{p}_i^{(t)})$, and $f(\mathbf{p}_i^{(t)})$ and $\mathbf{p}_g^{(t)}$.

Step 5: Repeat steps 2–4 until the stopping criteria is met e.g. when the number of iterations reaches the predefined $maxT$.

The training phase and classification phase of the proposed weighted ensemble of text classifiers are present in Algorithm 1, 2, and 3. The training process receives the inputs including the training text data $\mathbf{D}$, a set of K learning algorithms $\mathbf{K}$, feature type $\mathbf{F}$ (which are the types of feature we extracted from the text data $\mathbf{D}$, and $|\mathbf{F}| = S$), and some parameters of PSO (two popular parameters such as other population-based algorithms: maximum number of iterations $maxGen$, population size $nPop$ and three parameters of PSO: inertia weight: a, social and cognitive parameter C_1, C_2. In step 1-3, based on the feature type $\mathbf{F}$, we extract S features F_j from $\mathbf{D}$. In step 4-6, we train each of K learning algorithm $\mathbf{K}$ on the features F_j. The result of this training is the set of classifiers $\mathbf{H}$ including L classifiers.

In step 7-17, we generate the supports associated with each feature vector for the training observations. In detail, we apply T-fold Cross Validation on the training text data $\mathbf{D}$ to obtain the support matrix $\mathbf{S}$ which includes the supports of all classifiers for N training observations.

In step 18-19, we apply PSO to search for the optimal weights $\hat{\mathbf{w}}$ for combining classifiers. In Algorithm 2, for each candidate $\mathbf{w}$ generated in an iteration of PSO, we compute the combining result from the support for $\mathbf{x}_n$ i.e. $\mathbf{S}(\mathbf{x}_n)$ and $\mathbf{w}$ by using (2). By comparing the predicted label $\tilde{\mathbf{h}}(\mathbf{x}_n)$ and the ground truth label $\hat{y}_n$, we can calculate the $0-1$ loss value on each observation $\mathbf{x}_n$ (Step 4 in Algorithm 2). After looping through all training observations, we obtain the $0-1$ loss value on the training data associated with candidate $\mathbf{w}$. The value of the $0-1$ loss function will be used as the fitness value of each candidate. At each iteration of PSO, we update the position (i.e. the combining weight $\mathbf{w}$) of each particle using its previous position and its updated velocity vector (6) and (7). At the end of PSO, we obtain the optimal weight $\hat{\mathbf{w}}$ which has the lowest value of fitness function among all candidates.

The classification process works in a straightforward way (Algorithm 3). The process receives three inputs including the optimal weights $\hat{\mathbf{w}}$, the set of classifiers $\mathbf{H}$, and a sample $\mathbf{x}$ that needs to be classified. In the first step, $\mathbf{x}$ is classified by each of classifiers in $\mathbf{H}$. The output of this step is the support vector for $\mathbf{x}$ i.e. $\mathbf{S_x}$. The support vector is combined corresponding to the class label based on the optimal weight $\hat{\mathbf{w}} : \sum_{i=1}^{L} \hat{w}_{im} s_{i,m}(\mathbf{x})$. Based on the combining result, we assign a label to $\mathbf{x}$ by using (2).

IV. Experimental Setup

A. Datasets

To evaluate our method we used five benchmark datasets on text classification selected from different previous works [5], [6], [11], [27], [28]. A summary of dataset statistics is given in Table I. The number of classes ($|c|$) in each dataset is shown in column 2. We present the number of samples in training and testing set (t_r for train and t_s for test) in column 3, as well as the length of vocabulary from training set ($|v|$) in presented in column 4.

- **Newsgroup**: Newsgroup20 is a popular dataset with news articles from different sources categorized into 20

Algorithm 1 Training phase

```
Input: Training text data $\mathbf{D}$, learning algorithms $\mathbf{K}$, maximum number of generations: $maxGen$, population size: $nPop$, inertia weight: $a$, social and cognitive parameter $C_1, C_2$, feature type $\mathbf{F}$
Output: The optimal weights of $\hat{\mathbf{w}}$ and $\mathbf{H}$
    Extract Features
 1: for each type in $\mathbf{F}$ do
 2:     Extract feature $F_j$ from $\mathbf{D}$
 3: end for
    Generate the classifier
 4: for each pair of feature $F_j$ and learning algorithm $K_i$ do
 5:     Training the classifier $h_{ij}$
        $\mathbf{H} = \mathbf{H} \cup h_{ij}$
 6: end for
    Generate the support matrix
 7: Support $\mathbf{S} = \phi$
 8: $\mathbf{D} = D^{(1)} \cup \cdots \cup D^{(T)}, D^{(i)} \cap D^{(j)} = \phi (i \neq j)$
 9: for each $D^{(i)}$ do
10:     $D^{(-i)} = \mathbf{D} - D^{(i)}$
11:     for each feature $F_j$ do
12:         Get the data $F_j^{(-i)}$ and $F_j^{(i)}$ in $F_j$ associated with $D^{(-i)}$ and $D^{(i)}$
13:         Train ensemble of classifiers on $F_j^{(-i)}$ using each learning algorithm in $\mathbf{K}$
14:         Classify samples of $F_j^{(i)}$ by these classifiers
15:         Add outputs on samples in $F_j^{(i)}$ to $\mathbf{S}$
16:     end for
17: end for
    Search for optimal weights using the PSO method with $a, maxGen, nPop, C_1, C_2$
18: For each candidate $\mathbf{w}$, compute the loss value using Algorithm 2
19: Select the optimal $\hat{\mathbf{w}}$ at the end of PSO
20: return $\hat{\mathbf{w}}$ and $\mathbf{H}$
```

Algorithm 2 Compute the loss value for each candidate generated in PSO algorithm

```
Input: Candidate $\mathbf{w}$
Output: The loss value for $\mathbf{w}$
1: for each $\mathbf{x}_n \in \mathbf{D}$ do
2:     Compute the combining result from $\mathbf{S}(\mathbf{x}_n)$ and $\mathbf{w}$
3:     Assign a class label for $\mathbf{x}_n$ using hypothesis $\tilde{\mathbf{h}}(\mathbf{x}_n)$ (2)
4:     Compute $\mathcal{L}_{0-1}(\mathbf{w}, \mathbf{x}_n)$ by (3)
5: end for
6: Compute $\mathcal{L}_{0-1}(\mathbf{w})$ by (4)
7: return $\mathcal{L}_{0-1}(\mathbf{w})$
```

Algorithm 3 Classification phase

```
Input: Unlabeled sample $\mathbf{x}$, the optimal weights $\hat{\mathbf{w}}$ and $\mathbf{H}$
Output: Predicted class label for $\mathbf{x}$
1: Obtain the support $\mathbf{S}(\mathbf{x})$ by using $\mathbf{H}$
2: Compute the combining result from $\mathbf{S}(\mathbf{x})$ and $\hat{\mathbf{w}}$.
3: Assign the class label by using (2).
```

TABLE I
DATASETS USED IN THE EXPERIMENTS

Name	$\|c\|$	t_r/t_s	$\|v\|$
Newsgroup (NG)	4	8335/5558	44,333
Reuters (RU)	8	5485/2189	14,575
Deception (DC)	2	1200/400	9064
AG News (AG)	4	1.2M/7.6k	63,738
Dbpedia (DB)	14	5.6M/70k	727,621

different classes. For our experiment, we choose four major classes from Newsgroup20 dataset (comp, politics, rec & religion) following the approach adopted by [6].

- **AG News**: This is also a dataset comprised of news articles from more than 2000 data sources, out of which we select the four major classes with the highest number of samples following the approach adopted by [5].
- **Dbpedia**: This is a crowd-sourced community effort dataset with structured information extracted from Wikipedia. It contains abstract and topic from 14 non-overlapping classes obtained from Wikipedia articles [29].
- **Reuters**: This dataset consists of news articles from different categories and topics. We select the top 8 classes with the highest number of samples [30].
- **Deception**: The dataset consists of reviews collected from the users of 20 hotels in Chicago. It contains 800 truthful and 800 deceptive reviews. We use this dataset to classify the reviews into deceptive or truthful reviews [27], [28].

B. Experimental Settings and Benchmark algorithms

We selected 5 traditional machine learning algorithms namely Random Forest (RF), Naive Bayes (NB), Logistic Regression (LR), Support Vector Machine (SVM), and XgBoost (XgB) in constructing the ensemble. The configuration used for these algorithms are given as follows:

- **RF**: The number of trees was set to 100, the criteria for measuring the quality of a split was the Gini index.
- **NB**: Multinomial distribution of Naïve Bayes was used in which the smoothing parameter was set to 10.
- **LR**: The multinomial loss was used to fit across the entire probability distribution.
- **SVM**: The regularization parameter was set to 1 with squared L2 penalty and linear kernel.
- **XgB**: The number of estimators was set to 100 while the maximum depth allowed for a tree is kept to 3 by default.

These parameters were obtained using the grid search cross-validation on the train set of newsgroup data. We used the Scikit-Learn library [2] to implement these algorithms. For the feature representation, we used two features namely TF and TF-IDF to represent the text data for the traditional machine learning algorithms. For the training of PSO, the inertia weight a was set to 0.9 while two parameters c_1 & c_2 were set to 1.494. The number of iterations was set to 100 while the population size was set to 50.

[2]https://scikit-learn.org/stable/index.html

There have been a lot of developments in the field of text classification in terms of using DNNs. We compared our method with two deep-learning benchmark algorithms for text classification namely ULMFiT and BERT. ULMFiT [11] is one of the pioneer works proposing neural transfer learning for NLP. The authors presented a fine-tuning approach for sharing weights from a pre-trained Bi-LSTM LM for a classification model. This work out-performed many state-of-the-art algorithms on various text classification datasets. Following the success of transformers and ULMFiT, authors in [9] proposed a similar fine-tuning approach using transformers that out-performed many state-of-the-arts in various downstream NLP tasks.

The configuration used for these algorithms are:

- **ULMFiT**: Pre-trained AWD-LSTM LM [31] with highest learning rate allowed for pre-training as $1e-3$. Fine-tuning each layer with different learning rate starting from $2e-3/100$ to $2e-3$ [3] .
- **BERT**: Pre-trained LM with 12 transformer layers, each having 12 self-attention heads and 786 hidden layers feed-forward neural net. Bacth size for classifier training is 32 with learning rate as $2e-5$ and warmup proportion as 0.1 [4].

V. Results and Discussions

A. Ensemble with different configurations

We compared the performance of WEC with 5 different configurations, i.e. using different classifiers to construct the ensemble. Fig. 2 and 3 show the classification accuracy and macro F1 score of 5 versions of WEC on experimental datasets. It is noted that we could not run SVM on the 2 largest datasets, namely AG News and Dbpedia , because of the computational complexity. That's why observations shown in fig. 2 & 3 and table II & III don't have any value in AG News and Dbpedia results for combinations where SVM is involved.

Some of the observations from experimental results are as follows:

- Using different combinations of classifiers can obtain slightly different results. For example, WEC (RF, NB, LR) is about 2% better than WEC (RF, NB, LR, XgB) on the AG_News dataset.
- Among the different configurations of traditional machine learning algorithms, WEC obtains the best results on the Reuters and Deception dataset when RF, LR, NB, SVM, and XgB were used to construct the ensemble.
- When adding BERT and ULMFiT to WEC, the new ensemble is better than WEC with all the other configurations. For example, on the Deception dataset, the classification accuracy increases from nearly 88% to 90.5%.
- Similar patterns can be observed regarding the macro F1 score.

[3]Parameters taken from [11]
[4]Parameters taken from [9]

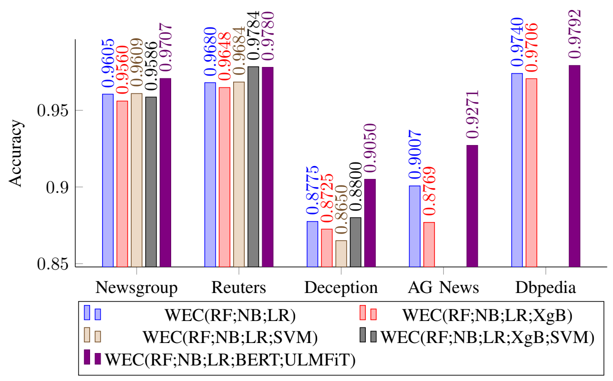


Fig. 2. Accuracy of WEC with different combinations.

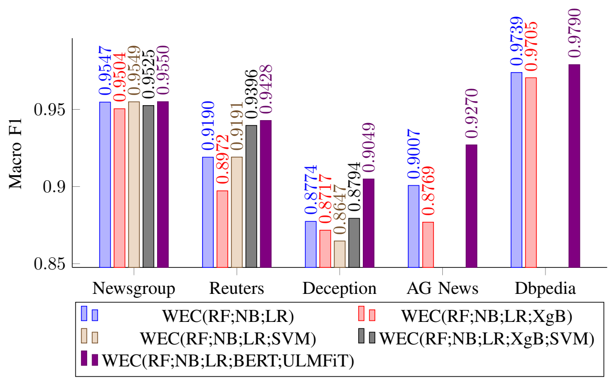


Fig. 3. Macro F1 score of WEC with different combinations.

The experimental results show the flexibility and robustness of WEC as this ensemble can be constructed with different classifiers to obtain high performance. Using only traditional classifiers obtain the acceptable performance with less resource usages. Meanwhile, using the combination between deep learning classifiers and traditional classifier can obtain the best results among all versions of WEC with high resource requirements. The selected combination for WEC is determined based on the specific situations. In the next section, we chose $WEC(RF, NB, LR, BERT, ULMFiT)$ to compare with the ensemble members.

B. Comparison with benchmark algorithms

Table II present the accuracy and macro F1 score while table III presents the ranking for 10 traditional machine learning classifiers, two deep learning classifiers, and WEC. Some of the observations from these results are given as follows:

- WEC achieves the lowest average rank among all methods (rank value 1.6 in terms of both accuracy and F1 score). On the five datasets, WEC ranks first on three smaller datasets, ranks second and thirds on the AG News and Dbpedia dataset.
- WEC is better than all constituent traditional classifiers. For instances, WEC is about 1.5% and 2% better than SVM and LR with TF-IDF, the second and third-rank method, on Newsgroup and Deception dataset.

TABLE II
THE CLASSIFICATION ACCURACY AND MACRO F1 SCORE OF ALL METHODS

						Accuracy							
Datasets	**RF TF**	**NB TF**	**LR TF**	**RF TFIDF**	**NB TFIDF**	**LR TFIDF**	**XgB TF**	**XgB TFIDF**	**SVM TF**	**SVM TFIDF**	**BERT**	**ULMFiT**	**WEC**
NG	0.9166	0.9530	0.9431	0.9188	0.9584	0.9559	0.8774	0.8792	0.9229	0.9571	0.9289	0.9429	**0.9707**
RU	0.9396	0.9652	0.9648	0.9465	0.9182	0.9588	0.9543	0.9492	0.957	0.9725	0.9584	0.9575	**0.9780**
DC	0.8675	0.8750	0.8450	0.8700	0.8650	0.8775	0.8225	0.8300	0.8300	0.8775	0.8675	0.7875	**0.9050**
AG	0.8747	0.8765	0.8853	0.8731	0.8777	0.8963	0.7898	0.7888	-	-	0.9259	**0.9501**	0.9271
DB	0.9636	0.9506	0.9722	0.9639	0.9455	0.9718	0.9324	0.9343	-	-	**0.9925**	0.9915	0.9792
						Macro F1							
Datasets	**RF TF**	**NB TF**	**LR TF**	**RF TFIDF**	**NB TFIDF**	**LR TFIDF**	**XgB TF**	**XgB TFIDF**	**SVM TF**	**SVM TFIDF**	**BERT**	**ULMFiT**	**WEC**
NG	0.9085	0.9465	0.9342	0.9105	0.9518	0.9486	0.8713	0.8732	0.9135	0.9508	0.9193	0.9332	**0.9550**
RU	0.7996	0.9160	0.9042	0.8413	0.6846	0.8664	0.9033	0.8960	0.8950	0.9386	0.9266	0.8939	**0.9428**
DC	0.8672	0.8747	0.8447	0.8699	0.8644	0.8773	0.8224	0.8300	0.8294	0.8773	0.8674	0.7863	**0.9049**
AG	0.8740	0.8758	0.8852	0.8725	0.8772	0.8961	0.7898	0.7888	-	-	0.9230	**0.9499**	0.9270
DB	0.9635	0.9504	0.9722	0.9638	0.9452	0.9718	0.9320	0.9334	-	-	**0.9920**	0.9915	0.9790

TABLE III
THE RANKING OF ALL METHODS

						Accuracy							
Datasets	**RF TF**	**NB TF**	**LR TF**	**RF TFIDF**	**NB TFIDF**	**LR TFIDF**	**XgB TF**	**XgB TFIDF**	**SVM TF**	**SVM TFIDF**	**BERT**	**ULMFiT**	**WEC**
NG	11	5	6	10	2	4	13	12	9	3	8	7	**1**
RU	12	3	4	11	13	5	9	10	8	2	6	7	**1**
DC	6.5	4	9	5	8	2.5	12	10.5	10.5	2.5	6.5	13	**1**
AG	8	7	5	9	6	4	10	11	-	-	3	**1**	2
DB	7	8	4	6	9	5	11	10	-	-	**1**	2	3
Avg. Rank	8.9	5.4	5.6	8.2	7.6	4.1	11	10.7	9.17	2.5	4.9	6	1.6
						Macro F1							
Datasets	**RF TF**	**NB TF**	**LR TF**	**RF TFIDF**	**NB TFIDF**	**LR TFIDF**	**XgB TF**	**XgB TFIDF**	**SVM TF**	**SVM TFIDF**	**BERT**	**ULMFiT**	**WEC**
NG	11	5	6	10	2	4	13	12	9	3	8	7	**1**
RU	12	4	5	11	13	10	6	7	8	2	3	9	**1**
DC	7	4	9	5	8	2.5	12	10	11	2.5	6	13	**1**
AG	8	7	5	9	6	4	10	11	-	-	3	**1**	2
DB	7	8	4	6	9	5	11	10	-	-	**1**	2	3
Avg. Rank	9	5.6	5.8	8.2	7.6	5.1	10.4	10	9.33	2.5	4.2	6.4	1.6

- WEC is better than ULMFiT and BERT on 3 small datasets. On Newsgroups dataset, for example, WEC is 4.5% and 3.5% better than BERT and ULMFiT, respectively. Although WEC performs poorly on AG News and Dbpedia datasets compared to BERT and ULMFiT, the prediction accuracy of WEC and the first rank-method is not significantly different (for example, 0.9271 vs. 0.9501 of WEC vs. ULMFiT on the AG News dataset).
- XgBoost is the poorest method in our experiment and its performance is by far worse than WEC. For example, on the Newsgroup dataset, XgBoost with TF obtains 87.74% accuracy whereas by using TFIDF, XgBoost obtains 87.92%, which is approximately 9% lower than the classification accuracy of WEC.
- A similar pattern can be seen in terms of the macro F1 score as WEC continues to rank first (with rank value 1.6), followed by SVM with TF-IDF.

TABLE IV
THE OPTIMAL COMBINING WEIGHTS FOR NEWSGROUP DATASET

Algorithms	$\mathbf{y}_1$	$\mathbf{y}_2$	$\mathbf{y}_3$	$\mathbf{y}_4$
RF TF	0.0426	0.5399	0.5987	0.0968
NB TF	0.3942	0.3242	0.9998	0.2307
LR TF	0.2262	0.7502	0.0297	0.1283
RF TFIDF	0.0255	0.6907	0.2336	0.0523
NB TFIDF	0.3255	0.4403	0.5811	0.1237
LR TFIDF	0.2328	0.3623	0.6504	0.1351
ULMFiT	0.4442	0.8876	0.7106	0.9463
BERT	0.9998	0.2307	0.3623	0.1351

Table IV shows the combining weights for the Newsgroup dataset obtained by solving the optimisation problem in (5) with the PSO. The weights reflect the contribution of each classifier on the combined result. For example, for the class $\mathbf{y}_1$, WEC algorithm gives less weights to the Random Forest (0.0426 & 0.0255) but more to BERT (0.9998). In the same way for class $\mathbf{y}_2$, higher weights are given to logistic regression TF and ULMFiT (0.7502 & 0.8876) compared to BERT (0.2307). In this way, WEC decides the contribution of each classifier in the final prediction.

The experimental results confirm the advantage of WEC compared to the traditional machine and deep learning classifiers, especially on the smaller datasets. In terms of time complexity, training time for WEC is slightly higher than other classifiers due to the PSO training required for weight optimisation. There is no difference in testing time of WEC compared to other classifiers.

VI. CONCLUSION

In this work, we presented an idea of the weighted ensemble of different text classifiers. Instead of using the majority voting mechanism for combining these classifiers, we propose a weighted combination approach in which the classifiers contribution to the collaborative class prediction varies. These weights were found by minimising the 0-1 loss function on the training data with the PSO algorithm. Extensive experiments were conducted using 10 machine learning classifiers and 2 deep learning classifiers and performance with respect to classification accuracy and macro F1 score was compared with constituent members of the ensemble. Our proposed method performed better than the state-of-the-arts on smaller datasets as well as better than all the baselines on each dataset. WEC achieves the lowest average rank among all other methods used and can be applied in real-world applications for high prediction performance. In a cold start scenario of a business process, where we have less labelled data for supervised downstream tasks, our method can be helpful in achieving better performance. In situations where it is hard to decide which learning algorithm or text representation should be used, our method provides an excellent option giving robust, effective performance across a range of datasets.

References

[1] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval.* Cambridge university press, 2008.

[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[3] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1532–1543, Association for Computational Linguistics, Oct. 2014.

[4] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1746–1751, Association for Computational Linguistics, Oct. 2014.

[5] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, pp. 649–657, 2015.

[6] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu, "Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, (Osaka, Japan), pp. 3485–3495, The COLING 2016 Organizing Committee, Dec. 2016.

[7] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, p. 2267–2273, AAAI Press, 2015.

[8] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, (New Orleans, Louisiana), pp. 2227–2237, Association for Computational Linguistics, June 2018.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[10] B. McCann, J. Bradbury, C. Xiong, and R. Socher, "Learned in translation: Contextualized word vectors," in *Advances in Neural Information Processing Systems*, pp. 6294–6305, 2017.

[11] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Melbourne, Australia), pp. 328–339, Association for Computational Linguistics, July 2018.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[13] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

[14] T. T. Nguyen, A. W.-C. Liew, X. C. Pham, and M. P. Nguyen, "A novel 2-stage combining classifier model with stacking and genetic algorithm based feature selection," in *International Conference on Intelligent Computing*, pp. 33–43, Springer, 2014.

[15] T. T. Nguyen, T. T. T. Nguyen, X. C. Pham, and A. W.-C. Liew, "A novel combining classifier method based on variational inference," *Pattern Recognition*, vol. 49, pp. 198–212, 2016.

[16] T. T. Nguyen, M. T. Dang, A. W. Liew, and J. C. Bezdek, "A weighted multiple classifier framework based on random projection," *Information Sciences*, vol. 490, pp. 36–58, 2019.

[17] G. Chen, D. Ye, Z. Xing, J. Chen, and E. Cambria, "Ensemble application of convolutional and recurrent neural networks for multi-label text categorization," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2377–2383, IEEE, 2017.

[18] R. Xia, C. Zong, and S. Li, "Ensemble of feature sets and classification algorithms for sentiment classification," *Information sciences*, vol. 181, no. 6, pp. 1138–1152, 2011.

[19] G. Wang, J. Sun, J. Ma, K. Xu, and J. Gu, "Sentiment classification: The contribution of ensemble learning," *Decision support systems*, vol. 57, pp. 77–93, 2014.

[20] A. Onan, S. Korukoğlu, and H. Bulut, "A multiobjective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification," *Expert Systems with Applications*, vol. 62, pp. 1–16, 2016.

[21] T. T. Nguyen, A. V. Luong, M. T. Dang, A. W.-C. Liew, and J. McCall, "Ensemble selection based on classifier prediction confidence," *Pattern Recognition*, vol. 100, p. 107104, 2020.

[22] T. T. Nguyen, A. V. Luong, T. M. Van Nguyen, T. S. Ha, A. W.-C. Liew, and J. McCall, "Simultaneous meta-data and meta-classifier selection in multiple classifier system," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 39–46, 2019.

[23] T. T. Nguyen, A. V. Luong, M. T. Dang, L. P. Dao, T. T. T. Nguyen, A. W.-C. Liew, and J. McCall, "Evolving an optimal decision template for combining classifiers," in *International Conference on Neural Information Processing*, pp. 608–620, Springer, 2019.

[24] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948 vol.4, Nov 1995.

[25] L. Zhang, Y. Tang, C. Hua, and X. Guan, "A new particle swarm optimization algorithm with adaptive inertia weight based on bayesian techniques," *Applied Soft Computing*, vol. 28, pp. 138–149, 2015.

[26] R. l. Perez and K. Behdinan, "Particle swarm approach for structural design optimization," *Computers & Structures*, vol. 85, no. 19-20, pp. 1579–1588, 2007.

[27] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock, "Finding deceptive opinion spam by any stretch of the imagination," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, (Portland, Oregon, USA), pp. 309–319, Association for Computational Linguistics, June 2011.

[28] M. Ott, C. Cardie, and J. T. Hancock, "Negative deceptive opinion spam," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (Atlanta, Georgia), pp. 497–501, Association for Computational Linguistics, June 2013.

[29] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[30] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European conference on machine learning*, pp. 137–142, Springer, 1998.

[31] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing LSTM language models," in *International Conference on Learning Representations*, 2018.